AD-A116 503 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA F/G 9/2
VIRTUAL INFORMATION FACILITY OF THE INFOPLEX SOFTWARE TEST VEHI--ETC(U)
MAY 82 P LU N00039-81-C-0663
UNCLASSIFIED M010-8205-11 NL

1 OF 2
AD A
116503

(12)

Center for Information Systems Research

Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts 02139

82

12

Contract Number N00039-81-0663 (MIT # 91445)
Internal Report Number M010-8205-11
Deliverable Number 6

VIRTUAL INFORMATION FACILITY
OF THE INFOPLEX SOFTWARE TEST VEHICLE
(PART II)

Technical Report #11

BY

Peter Lu

May, 1982

Principal Investigator:
Professor Stuart E. Madnick

Prepared for:
Naval Electronics Systems Command
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report #11	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Virtual Information Facility of the INFOPLEX Software Test Vehicle (Part II)	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER: M010-8205-11
7. AUTHOR(s): Peter Lu	8. CONTRACT OR GRANT NUMBER(s): N0039-81-C-0663
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Sloan School of Management, MIT 50 Memorial Drive, Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: May 1982
	13. NUMBER OF PAGES: 144
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

database computer, database management system, Software Test Vehicle, Virtual Information hierarchical system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the software design and implementation of the rear-end for the Virtual Information Facility of the INFOPLEX database computer. It is part of a major effort to develop a software simulation, STV, for the underlying architecture of INFOPLEX.
The virtual information facility is a single level of operations situated within the Functional Hierarchy. It supports the use ofvirtual information, a virtual entity based on procedural relationships and derivations from

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

physically recorded data. Upon completion, this facility will be integrated within the current implementation of the STV for the INFOPLEX Functional Hierarchy which lacks the support for virtual information processing.

Virtual Information Facility
of the INFOPLEX Software Test Vehicle

by

PETER LU

Submitted to the Department of Electrical Engineering and Computer Science in May, 1982, in partial fulfillment of the requirements for the degree of Bachelor of Science

Abstract

This thesis is a software design and implementation of the rear-end for the Virtual Information Facility of the INFOPLEX data base computer. It is part of a major effort to develop a software simulation, so called a Software Test Vehicle, STV , for the underlying architecture of INFOPLEX.

INFOPLEX is a hierarchical architecture for data base computers, based on functional decomposition of data base operations. It is a current research project of the Information Systems Group at M.I.T.'s Sloan School of Management. Within the INFOPLEX architecture, a functional hierarchy of information management functions is built on top of a storage hierarchy of information storage functions. These two independent hierarchies are further divided into many sub-levels, each of which is devoted to a more specific function of data base activities.

The virtual information facility is a single level of operations situated within the functional hierarchy. It supports the use of virtual information, a virtual entity based on procedural relationships and derivations from physically recorded data. Upon completion, this facility will be integrated within the current implementation of the the STV for the INFOPLEX functional hierarchy which lacks the support for virtual information processing.

Thesis Supervisor: Professor Stuart E. Madnick
Sloan School of Management, M.I.T.

Contents

1.0.0 INTRODUCTION

INFOPLEX DATA BASE COMPUTER is a current research project of the Information Systems Group at M.I.T.'s Sloan School of Management. It proposes a new architecture whose objectives are to provide substantial improvements in information management performance over conventional computer architectures, and to provide highly reliable support for very large and complex data bases.

1.1.0 INFOPLEX OVERVIEW

Progress of modern society has put increasingly more new and challenging demands upon the capability and performance of information storage, retrieval, and management. Conventional computers, whose architecture is designed primarily for computational objectives, are not suited to meet the requirements of these new demands. Efforts have been made in four different areas to build computer systems which will suit our information needs today, and in the future: (1) new instructions through microprogramming, (2) intelligent controllers, (3) dedicated computers for data base operations, and (4) data base computers. INFOPLEX is a research project belonging to the fourth category.

1.1.1 CONCEPT

INFOPLEX employs the concept of hierarchical decomposition which organizes information management functions into a functional hierarchy, and the physical memory management functions into a storage hierarchy (Madnick 78); both hierarchies consist of many independent levels of operation, each of which supports a different set of information or storage management functions through the use of multiple microprocessors.

1.1.2 INFOPLEX ARCHITECTURE

As stated previously, INFOPLEX is an architecture for data base computers based on hierarchical decomposition. A functional hierarchy of information management functions is built on top of a hierarchy of information storage functions. Both hierarchies are further divided into many functionally independent levels of operation, each of which is to be supported by a set of micro-processors operating in parallel with one another. A global Communication Bus coordinates inter-level transmission of data. This hierarchical architecture exploits the advantages of functional modularity of operations, and of parallel processing of micro-processors to systemize data base activities and to achieve a prescribed level of efficiency. A graphical illustration of this architecture is presented in appendix A (1.1.2).

1.1.3 FUNCTIONAL HIERARCHY

Current architecture of the functional hierarchy (Hsu 1982) with respect to data abstraction consists of four separate levels: (1) external level, (2) conceptual level, (3) entity level, and (4) internal level. A part of the conceptual level is a virtual information facility (Hsu 1982). These four levels of information management are highly independent of one another, and each is responsible for a different but necessary phase of information processing in a data base computer.

1.1.4 RESEARCH ISSUES

Major efforts of INFOPLEX research are devoted to the design, modeling, and evaluation of an optimal decomposition strategy for both the functional and memory hierarchy of information management and storage operation, and also to the study of an associated distributed control mechanism. This control mechanism would be used to coordinate the activities of and inter-level communications within the hierarchies.

1.2.0 THESIS OBJECTIVE

This thesis shares a joint mission with a concurrent thesis by Jameson Lee. The two theses are entirely separate in functionalities, but closely related and dependent upon one

another for a complete software simulation of the virtual information facility on the INFOPLEX data base computer architecture. This facility would incorporate the design and implementation of two sub-levels of the INFOPLEX functional hierarchy, the virtual information level, and an user interface level which is tailored for the use of virtual information processing.

Jameson Lee's thesis is responsible for the fulfillment of the front-end objectives of the joint mission; his objectives include the design and implementation of the following:

a) A data base language to support virtual information

b) A finite state machine to parse data base statements written in this language

c) A user-interface tailored to the use of virtual information.

d) A processor to process the creation, listing, and modifications of virtual definitions, as well as the substitution of these definitions into data base statements in actual use.

This processor would also be responsible for transforming data base statements into a chain of tokens, each of which

would include an indicator describing the classification of the token according to a prescribed classification scheme.

My objectives involve the implementation of the following:

a) An execution tree of expressions and conditions within retrieval statements.

b) A parser which implements a finite state machine that takes the match-action-next_state rules as input from the front-end.

c) An entity set table of real and virtual entity sets within retrieval statements.

d) An interface with the entity level in regard to the passing of values of real attributes.

e) An extraction of the data which the user specified in his/her database statements.

f) A virtual definition catalogue which consists of two separate dictionaries, permanent and adhoc.

The combined objectives of my "back-end" and Jameson Lee's "front-end" would fulfill our joint mission as mentioned ear-

lier, namely, to construct in software a virtual information facility with its own user interface, from here on referred to as VIFI, the Virtual Information Interpreter.

1.2.1 BACKGROUND

In the three short months in which VIFI was develped, we labored and wished to exhibit a certain degree of professionalism in its design and implementation. The merits of modular programming, of innovative algorithms, of performance efficiency, of functional capabilities, of user-friendliness of the proposed data based language, of program organization and flexibility, and even of consistencies in programming style were evaluated against time and labor limitations. A serious attempt was made to incorporate all of these characteristics into our Virtual Information Interpreter.

2.0.0 VIRTUAL INFORMATION

2.1.0 Concept

The concept of virtual information in data base systems has been developed and examined in earlier research of the Information Systems Group. Basically, there is a spectrum of the kinds of information which may be retrieved from a data base. Along this spectrum, pure data occupy an extreme on one end, and pure algorithms occupy the extreme on the other. In between these two extremes are the information which may be derived from a combination of data and algorithms; such information are dynamic and procedural in nature, and are referred to as Virtual Information.

2.2.0 CLASSIFICATION

Virtual information may be categorized into three major classes: factored facts, inferred facts, and computed facts. Together, these three classes of virtual information and combinations there of, constitute the portion of the information spectrum between the two extremes of pure data and pure algorithms.

2.2.1 FACTORED FACTS

Factored facts, subsets of data elements, based on certain prescribed conditions, or so called predicates, of attribute values, are often very valuable in structuring information in a useful manner. For instance, if a certain data base maintains records of weight, hair color, and salary for a group of employees, it may be useful to select from this group those individuals who share a certain condition on their attribute values, such as having black hair, making a salary greater than 8 dollars per hour, or weighing over 300 pounds. It is important that users of information should be able to access information independent of the particular factoring involved; this would imply the ability to support multi-level factoring, or repeated factoring of data.

2.2.2 COMPUTED FACTS

Computed facts are those information which are obtainable through the application of particular computational algorithms and operators on data or groups of data. These operators include arithmatic, comparative, boolean, and other kinds of functions. In the very least, computed facts include those pure data manifested in a different form, with a different unit of measure, or an alias name. For instance: a user may define a virtual age attribute to be the difference between the current year and a person's birth-year, a virtual rectangular area attribute to be the length multiplied by the width, or an

attribute value in the unit of inches to be 12 times the attribute value in the unit of feet. In this sense, transformations between different units of measure are intrinsic to the operations of computed facts.

2.2.3 INFERRED FACTS

Inferred facts pertain to implicit relationships which the data base system may arrive at through certain levels of indirection. In other words, a path, although indirect, does exist which leads to the desired data in storage. There are two ways by which the system on its own can support this kind of virtual information. The first method is by an exhaustive search of all possible paths, and the second is the application of a certain degree of artificial intelligence to deduce a viable path to the target data. Well, the first method is unbounded in computing time, and even when a path is found, it may not be the correct path; the second method is far fetched at this time. Therefore, we will give our attention to a different but comparable set of inferred facts which is implementable, and we give it the name Pseudo Inferred Facts. Pseudo Inferred Facts are exactly the same as inferred facts except that all the indirections will be explicitly designated by the user. With this strategy, exhaustive search is not necessary, artificial intelligence is not necessary, and the specified path would always be the designated and correct path. For instance, the Uncle relationship may be defined as the application of the

Brother relationship after the application of the Mother relationship.

2.3.0 SPECIFICATION

Users of information, through the virtual information facility, define their own working environment and the manner in which they would like to use the physical and underlying data. Such definitions of virtual information may be accomplished through a virtual information definition language. The virtual information facility would accept virtual information definitions and their modifications in the definition language, and respond to virtual information retrieval requests through a separate virtual information retrieval language.

2.4.0 MERITS

There are several major merits in the support of virtual information in a data base system. It is dynamic in nature because its definition may be created, deleted, and modified readily; its definition applies to all instances of data where it may apply, and yet there is but only one copy of this definition stored in the system. By facilitaing the ease of modification, it enhances data base flexibility, by eliminating redundant physical records, it contributes to more consistent data, and by being procedural in nature, it enhances

information accuracy through the delay in the evaluation of data which vary over time or other changing factors until their time of use. These kinds of merits are based on virtual information's association with procedural relationships. For instance: the stored algorithm for computing age would eliminate the need to update the age attribute day by day if it were physically stored, and would be applied to calculate anyone's age, thus eliminating redundancy of stored information.

Virtual information also conserves the use of vast amounts of physical storage. It makes unnecessary the storage and maintainence of those information which may be derived upon request. This raises the issue of Time/Space trade-off, which should be seriously considered when deciding which kinds of fundamental data are or are not to be physically stored. Derivation upon requests will have the added cost of derivation; therefore, those information which will be used many times and are also difficult to derive may be the best kind of data to be physically stored; those information which is seldomly used and easy to derive may be the best kind of data not to be physically stored. Furthermore, the situation is made even more complex as we realize that the definitions themselves will require the use of physical storage. Thus, it wouldn't be an easy task to decide which kinds of data are to be derived, or to be actually stored.

The definition of virtual information on a per user basis would simulate an entire virtual data base for each individual user. Each one would be free to tailor the data base to his own preferred view or use through the virtual information definitions. A particular set of virtual definitions may be very useful for one group of users, and another set for another group of users. In this sense, each one has gotten a data base suited for his own use while not affecting anybody else's usage of the data base. A logical extension of this scenario is to implement access control mechanisms such that users may establish a controlled sharing of sets of virtual information definitions with one another; the data base administrator may monitor all such sharing to prevent unauthorized access to a certain set of virtual information functions. However, in a scenario as such, a separate catalogue would have to be maintained for each and every user, and considerable catalogue management would be required. Such is the cost for this individually user-tailored data base functionality, a secondary merit of the use of Virtual Information.

2.5.0 APPROACH

The concept of virtual information leads directly to a functional approach to data bases. A virtual information facility would be treated as a collection of functions, and retrieved data would be regarded as functional values. Virtual information requests correspond to function invocations; this func-

tional approach to information readily supports procedural relationships on which based the concept of virtual information. As a result, a virtual information facility is likely to resemble very much a language interpreter which accepts functional definitions and responds to functional invocations with specified arguments.

3.0.0 FUNCTIONALITIES

There are numerous functionalities to a virtual information facility, each of which may be implemented to a varying degree of completeness. Although it may be desirable to implement all the functionalities there are wherever possible, it may be too impractical and less than meaningful for the initial version of the implementation. Thus, we have not implemented the One Data Base per user feature of virtual information capabilities which we have described in the previous chapter. Later portions of this chapter would describe the functionalities of virtual information which we did implement; surely, not all of these implementations would be without room for further refinement, even though they already include an extensive set of virtual information capabilities.

3.1.0 UNDERLYING DATA MODEL

The virtual information facility lies on top of the entity set level of the functional hierarchy. In this level, the data base is seen as a network of entity sets and their attributes. Each entity set may have a varying number of attributes, some of them being value attributes and others being entity attributes. (Hsu 1980) The value attributes include a set of attribute values, and the entity attributes represent

relationships leading to other entity sets. Illustrations are found in appendix A (3.1.0).

3.2.0 ACTIVE WORKSPACE

We have developed an active workspace which incorporates a line editor with full screen display, through which user commands may be issued. The workspace consists of two buffers, an execution buffer, and a transaction buffer. The transaction buffer holds many data base statements which will be executed sequentially when the transaction buffer is executed. The execution buffer holds a single data base statement and will be automatically executed when a data base statement is completed. A number of buffer commands is created to manipulate buffer contents. Examples of these commands as well those of the data base statements are illustrated in Appendix A (3.2.0) since their design is detailed in the front-end description of the VIFI.

3.3.0 PERMANENTLY DEFINED VIRTUAL INFORMATION

Permanent virtual information may be defined through the Define statement. Such definitions will be stored in a global dictionary, or catalogue, in the form of character string, and will remain there until explicitly removed or over-written by a different definition. Examples may be found in appendix A (3.3.0).

3.4.0 ADHOC VIRTUAL INFORMATION

Virtual information definitions may be derived for only the duration of a single transaction. When all statements within the transaction are executed, the adhoc dictionary would be erased. Within the transaction, adhoc definition may be created, deleted, as well as modified at any time. With this feature, each transaction would be associated with a catalogue of its own, and would not interfere with the concurrent activities of other transactions executing in parallel. At this stage, we do not support concurrent transactions, but adhoc definition capability is still useful in the principle of individual environments of transactions and the scoping of virtual variables. Naturally, the permanent dictionary would also be accessible from within each transaction.

3.5.0 NOTION OF A TRANSACTION

A transaction is a body of executable statements joined together within a single context. This context is provided by the adhoc dictionary associated to the particular transaction. A transaction is created within the transaction buffer, and will remain there until it is explicitly over-written, erased, or executed. Merits of this transaction concept are threefold: a) a group of statements which collectively does a certain task may be consolidated to exhibit logical unity. b) a shared context may be created and maintained for each transaction, a sign

of transactional modularity and independence from one another.
c) the execution of the consolidated operations in a transaction may be put off until a more opportune moment, by which time new permanent or adhoc virtual information definitions may be defined either to supplement or to replace existing definitions.

3.6.0 VIRTUAL ATTRIBUTES

Virtual attributes equated to the results of computational algorithms acting on available data or of designated indirect references may be explicitly defined through the Define data base statement. This feature incorporates the support for Computed Facts as well as for Pseudo Inferred Facts. For instance, the following is the definition and usage of two virtual attributes, income and ship-country, a computed fact, and a pseudo inferred fact.

```
Define income as salary - expenses ;
Retrieve ( {teachers} ) by ( {V0} name,income) ;
```

The foregoing retrieve statement returns two vertical columns of data. The first column being teacher's name, and the second column being their corresponding incomes.

```
Define ship-country as company ( country (name)) ;
Retrieve ( {ship} ) by ( {v0} name, ship-country) ;
```

This foregoing retrieve statement returns two columns of data, the first being individual ship names, and the second being the name of the country to which the ship belongs. More examples of this entity set is in appendix A (3.6.0).

3.7.0 CONDITIONS ON REAL OR VIRTUAL ATTRIBUTES

Arbitrary conditions on real or virtually defined attributes may be defined by INFOPLEX users as the shared 'condition' on their data values from which factored facts may be later constructed. For example:

```
  Define old as age > 70 ;
  Define rich as assets > 1000000 ;
  Retrieve ( {people} where (rich and old))
 by ( {VO} name);
```

The foregoing retrieve statement would return a list of names of those people whose age > 70 and assets > 1000000.

3.8.0 VIRTUAL ENTITY SETS

Aside from virtual attributes, we also support a basic notion of virtual entity sets. We recognize two kinds of virtual entity sets:

a) Union, intersection, or cartesian of real or previously defined virtual entity sets based on their real and virtual attribute values.

b) Subsetting of real or virtual entity sets based on certain conditions on their real and virtual attribute values.

For instance:

Define ClassAB as {{ClassA} MU (Name) {ClassB}} ;

ClassAB is defined as the result of a multiple-union operation on entity sets ClassA and ClassB, based on a common attribute called Name.

Define RichMen as {Men} where (assets > 1000000) ;

RichMen is defined as a virtual subset of the set Men, based on the values of its asset attributes.

The complete set of union and intersection operators as well as the cartesian product operator between entity sets is illustrated in appendix A (3.8.0). Also included are details of the capability to specify various conditional predicates on attribute values.

3.9.0 GENERALIZED MACRO FACILITY

Users will be able to give arbitrary definitions to specified names and late substitute for these names in the database statements. In this sense, the define statement may be used not only to define virtual attributes and virtual entity sets, but also random definitions that may seem to be incoherent without the proper context. When a retrieval statement is to be executed, each word within the statement is checked against a list of stored definition names; any word that matches with any definiton name would be replaced from the dictionary by the definition. Furthermore the words in the definition substuted would likewise be substituted, if possible.

3.10.0 EXTENDABLE FUNCTIONALITIES

3.10.1 USER DEPENDENT VIRTUAL DEFINITIONS

This particular functionality is not difficult to implement, but it may be unnecessary at this stage of the project. It simply would require a separate catalogue for each user which includes an access control list, proper search rules including default situations, and adequate coordination and control mechanisms to manage the various catalogues. It would increase the cost in terms of time and space efficiency. Thus, we have not included this functionality in this version of virtual information implementation. Nevertheless, if circumstances in

later time are such that the support for user dependent catalogues is so desirable as to more than compensate for its cost of implementation, this functionality may be added readily. At this time, we can simulate this by tagging all definition names with the unique user id of the user and hence make the name unique.

3.10.2 INFERRED FACTS OF UNDESIGNATED INDIRECTION

Inferred facts with undesignated indirection, rather than pseudo inferred facts with designated indirection, is likely to have tremendous costs in system performance whenever it is to be implemented. As previously stated, this would require either an exhaustive search or a certain level of artificial intelligence, both of which require large amounts of resources in computing power, storage and time. Furthermore, in order to verify that the indirection the system chooses at each step along the way is correct, the user has to monitor the computer decisions interactively; this defeats the original purpose of not having the user to designate his intended path of indirection. Thus, it seems very doubtful that this functionality will ever be implemented unless the requirements for user monitoring of the decision process is somehow eliminated.

4.0.0 OVERVIEW OF VIRTUAL INFORMATION SUB-SYSTEM

4.1.0 DIVISION BY INFOPLEX SUB-LEVELS

The virtual information sub-system related to our project extends into two levels of INFOPLEX, due to necessity from the modular designer's point of view. The interaction between the user and the input peripherals of the sub-system must be isolated to one level, seperate from that used to process the requests. Since part of our sub-system resides in a level external to the Virtual Information Level, data flow must be channeled through the INFOPLEX control structure data bus when passing to and from the input section to the processing section, in a proper implementation. A diagram of our entire sub-system and the surroundings is in appendix A (4.1.0).

4.1.1 USER INTERFACE LEVEL

The user interface level currently consists of a user interface activity coordinator and a virtual information interface buffer. When completely and correctly implemented, the user activity coordinator acts as a switchbox to activate either the virtual information interface buffer or a real information interface buffer. Likewise, it arbitrates between sending the input data to the virtual information processor or the real

information process in the Virtual Information Level. In addition to placing user inputs into proper storage, the interface buffer alters certain characters from the input so that the lower levels may safely use those characters as delimiters.

4.1.2 VIRTUAL INFORMATION LEVEL

This level is partitioned into two sections, one of which is the real information processor that creates and updates entity sets. We are not concerned with that portion of the design. The section of interest is the virtual information processor. Here we have a virtual information activity coordinator that sequences the data manipulators. First it has the input statements cleaned of virtual information by substitution from the dictionary; it then converts the statements into tokens. The tokens translate into database query commands when they are parsed using finite state transition rules. The coordinator then sends some queries to the lower INFOPLEX entity level for processing. The returned data are relocated to useable storage and processed further to extract the exact elements which the user specified by his/her command. Control then goes back to the User Interface Level.

4.2.0 INTERNAL INTERFACES

To avoid unnecessay conflicts and lack of determinism in data transfer between routines internal to a level, the coordina-

tors in the two levels, User Interface and Virtual Information, are capable of monitoring all data movements. This is a merit of the horizontal structuring of data accessing and processing paths. Each interface is well defined. The user interface coordinator transfers character buffers in both directions to the buffer processor. The buffer can be then altered minimally and sent down the INFOPLEX bus. The virtual information level activity coordinator sends a facsimile of the character buffer to the tokenizer and receives a list of tokens in return. The virtual level coordinator then passes the tokens to the parser and receives a set of filled tables which are copied into a simplified version to be used as a query request to the lower levels. The filled tables become the main means of transferring data from then on. There presently exists a dictionary which should be later fully embedded in the database storage system. In other words, we treat internal interfaces very much like the external ones which attach to the INFOPLEX bus, since we want modularity and simplicity of operation.

5.0.0 STORAGE ARCHITECTURE AND MANIPULATING METHODS

Since the data retrieval requests and returns are based on structures, a necessity exists to clarify what is involved in accessing these structures and to exhibit the kinds of data manipulation that can be performed on the structures.

5.1.0 INTER-LEVEL QUERY STRUCTURES

5.1.1 ENTITIES

The structure ENTITY consists of thirteen sets of vertically arranged entity set representations. Each entity has a NAME, DEPTH (number of values in a single occurrence attribute), VES_FN (entity set function), WHERE (discrimination tree pointer), N_PARENT (entity parents pointers), VES_PAR (virtual entity set pointer to attribute map between parent entities and child entity), and a set of fifteen attributes. Each attribute has a VES_KEY (designates a key attribute to a virtual entity set), CART_KEY (designates a cartesian key attribute to a cartesian virtual entity set), SING_OCC (designates a 1:1 correspondence attribute), A_PARENT (assigns attribute parent if non-zero), USES (attribute name), and LIST (locates all values of an attribute).

Real entity sets are created from the information available in the entity set representation. An abridged version of ENTITY is made and sent to the entity level of INFOPLEX and is returned with attribute values linked to it.

Virtual entity sets are created from real entity sets or from existing virtual entity sets. Whenever VES_FN specifies that an entity set is virtual, the N_PARENTs have their attributes mapped by the N_MAP attached to VES_PAR. Based on the proper relationship amongst the keys and labels, N_MAP determines which attributes are generated by which existing attributes; this map is extremely useful for future linking of attributes across entity sets. The VES_FN defines the particular scheme for creating the entity set.

5.1.2 ATTRIBUTE VALUES AND LEVELS

The ATTRIB structure of itself is very simple. It contains one attribute value and the level number of that value. The inter-relations of the ATTRIBs is flexible but complex. Attribute values are linked across the attribute table in the entity set by the level number; such a set is always copied or skipped as one since the correspondence is entire. An example can be seen in appendix A (5.1.2).

5.2.0 INTER-LEVEL CONDITION STRUCTURES

5.2.1 EXECUTION TREE

XTREE is structured such that it represents a tree with nodes and links. Traversal through the tree gives a natural sequence of performing operations and hence a correctly built tree can be used as the control structure of data processing. Each node uses the LABEL field once and each arc of a node uses one LINK. The number of links in a node is stored in CHILD so that tree traversal is easily achieved.

5.2.2 TRANSITION RULES MACHINE

MACH stores the information that is necessary to transform database statement tokens to query structures. In a full implementation of INFOPLEX, MACH should be totally static, since it is needed for any inputs to the VIFI. The control structure itself should boot up the entering of information into this table.

5.2.3 ADDITIONAL INFORMATION TABLE

Useful for storing more information than is allowed by a single node in XTREE, and as a workspace to generate attributes, XCHNGE should reside in the VIFI as a local storage area, in the same repect as ENTITY.

5.3.0 EXTRA-LEVEL QUERY STRUCTURES

5.3.1 QUERY REQUESTER

RETE_ARG is basically an extracted version of ENTITY, with control structure information added. Additionally, it has a COND sub-structure to store simplifiable conditions for lower level processing.

5.3.2 QUERY INFORMATION RETURN

RETE_RTN is essentially the storage carrier for ATTRIB. It contains control structure information.

5.4.0 PSUEDO EXTRA-LEVEL QUERY STRUCTURES

There is currently storage used to simulate storage that should be available from within the database itself. This is due to the fact that simulated retrievals are more easy to handle and check for errors.

5.4.1 THE DICTIONARY

The DICTION table is split into three sub-tables, to simulate a number-of-lookups CHECKer, an ADHOC dictionary, and a MAIN dictionary. With appropriate tagging and catalogging of definition names, the main dictionary should eventually be able to

take on this task. Certain amounts of security restriction can be made by this dictionary interface as well.

6.0.0 SOFTWARE IMPLEMENTATION

The software described is fully listed in appendix B. We used PL/I on the IBM 370 machine, CMS operating system. Only those rear-end routines are described; front-end implementation can be found in Jameson Lee's documentation. By relating to the overview diagram in appendix A, one can see the correlation of the software routines.

6.1.0 DICTIONARY

The dictionary routine, DCTNRY, is used only to simulate the storage of data in the main database. Since some necessary functions, such as template string matching or variable length strings, are not yet available in the database, a more powerful simulator was substituted. Due to the modularity of the system, it would be trivial to replace this with the real thing.

Our dictionary currently has a few special features. It can do pseudo parameter passing by replacing characters in the definition by characters in the definition name matching string. It has a counter that prevents unlimited circular definitions. Most of all, it has a defined search order that consists of the built-in function catalogue, the adhoc dictionary, and the main dictionary. The adhoc dictionary and the

counter can be cleared by a special command generated by the dictionary user.

6.2.0 MACHINE DEFINER

DEFMCH simply reads the transition rules from an input file into a table which can be accessed more readily by indexing.

6.3.0 LANGUAGE PARSER

The PARSE routine basically interpretes the linked list of tokens specified by TOKENS_PTR and produced by the TKNIZE routine, and translates this list to a request format to be later simplified and sent to the lower entity level for processing and retrieval. It does the interpretation based upon a set of finite-state transition rules that have been input by the DEFMCH routine into a transition rules table named MACH. The parameters filled by the parser are the execution order tree, XTREE, the additional information exchange table, XCHNGE, and the unabridged entity set query table, ENTITY. Of course, the PARSE routine can be put into a DEBUG mode to follow the transition states that are passed given a set of tokenized retrieval statement and the transition rules table.

The main data storage elements in PARSE in addition to the parameters aforememtioned are the simulated stacks, STACK, and the virtual entity set map table, VES. The first two stacks are

used by the finite state parser in a push-down automaton scheme, in which they are respectively labeled as the operand stack and the operator stack, whereas the third one is used internally and exclusively by the parser to track the addresses to the execution tree.

A word of clarification is necessary here. In the description of PARSE to follow, the term 'item' refers to a multi-faceted abstraction of data in the sense that it could be anything from data on the level relevent to our processing, the same data with internally used update sources or labeling tags imbedded, to internally used mapping link numbers. Furthermore, the data being processed may be utilized by different levels, such as the finite automaton or the entity set request processor, in a related but differing way. For instance, the the item '+:B', which means 'plus of type built-in fuction', is the composition of a symbol on the automaton level typed in by the user and a internally generated tag; the '+' portion is used by the automaton level as tokens to be matched and by the entity set retrieval processor as the operation 'plus' to be performed.

PARSE has a main driver loop which is traversed once for each transition state that is entered by any particular input statement. The loop coordinates the pairing of matches that fire according to MATCHING and the actions to be used and the next

transition state to be entered according to ACTING. The top level sub-routines of PARSE are:

MATCHING which determines whether the states of the input linked list and the stacks are matchable with any of the required conditions for firing in a given transition state.

ACTING which assigns the actions taken by the current transition state to arrive at another state, and designates this next state.

GETS which is a general string functions that breaks an argument item into two parts seperated by a chosen character in the item.

PUSH which pushes an item onto a specified stack.

POP which pops the top item off a specifed stack.

CHANGE which alters an item based upon special characters in the item selecting sources of update.

GENENT which takes items off the stacks and generates an entity request into ENTITY.

VIRTX which replaces the top item on the operand stack, which must be a virtual entity abstract map number, with the corresponding real map number.

VIRTA which adds a new entity map number into the virtual entity set catalogue.

GENNODE which takes items off the stacks and generates a node on the execution tree, with the appropriate links to other nodes in the same tree. The link to the new node is put onto the operand stack.

GENATTR which puts the specified attribute name into the entity set retrieval table, with the appropriate links to other attributes in the particular entity set of interest. The specified attribute name is converted to a reference number to the attribute in the entity set table.

EXCH which exchanges the top item on the operand stack with the map number to the next available space in the additional information table so that further items may be appended to the additional information table as part of the current top item. A tag is appended to the top item to identify what type of additional information, indirection of attributes or multiple comparison, is at issue.

ADDON which appends additional items onto the most currently addressed space in the additional information table.

Of these, MATCHING, ACTING, GENNENT, GENNODE and GENATTR are complex enough to deserve furthur explanation.

MATCHING simply decides whether or not a certain transition state conditions item matches the current existing conditions of inputs. The conditions item may be a set of any number of conditions each of which if matched will flag the match as positive. Each condition specifies what must be found in a set of sources consisting of the first item on the input linked list, and the top of stacks of the first two stacks. Furthermore, allowance has been made to allow translation of a single matching item to a multiple list of possible matching items to effect storage minimization in the transition rules table.

ACTING processes the list of actions found in the transition rules table. It takes the actions one by one and distributes the processing over several loops. Each action consists of an action name followed by arguments whose existence and typing vary dependent on each individual action. A short summary of the actions consists of:

'push' which pushes its second argument onto the stack specified by the first.

'pop' which pops off the top of the stack specified by the first argument, using POP.

'del' which eliminates the first item on the input linked list by advancing TOKENS_PTR and deallocating the first token.

'genent' which invokes the sub-routine GENENT to generate an entity set.

'virtx' which replaces the top item on the operand stack with the real map number indicated by the item, using VIRTX.

'virta' which adds a new entity set mapping into the virtual entity set catalogue, using VIRTA.

'attwhr' which sets the appropriate linkage between the entity set table, ENTITY, and the execution tree, XTREE.

'gennode' which generates a node on the execution tree, using GENNODE.

'indx' which exchanges the top item on the operand stack with the map number to the next available space in XCHNGE, using EXCH with the tag for indirection.

'mulx' which exchanges the top item on the operand stack with the map number to the next available space in XCHNGE, using EXCH with the tag for multiple comparison.

'addon' which adds the top item on the operand stack onto the additional information table, using ADDON.

After all actions have been performed, ACTING designates the next transition state to be entered, based on the next state entry in the rules table, which if negative would indicate a 'subroutine return' found on the operator stack.

GENENT does the generation of entity set requests using the ENTITY table. The appropriate items must already exist on the stacks to allow for the generation and hence the parsing rules must provide for the set up. The types of entity sets are:

'r' which is an explicitly user defined entity set refering to a real entity set existing inside the database.

's' which is an implicitly created entity set that refers to a composite entity set based on a single entity set, with additional constraints attached.

'mu', 'mi', 'su', 'si', 'cs' which are explicitly user defined entity sets refering to composite entities each of

which is based on two entity sets. The specifics of each entity type are described elsewhere in this document.

The necessary attribute keys in any key lists are catalogued by GENENT. An entity set real map number is also placed on the operand stack at the conclusion of a generation to allow proper linking of entity sets.

GENNODE generates nodes on XTREE by taking the top item on the operator stack and reserving sufficient number of locations on the tree for the children of the operator of interest. It puts the link numbers to these locations on the internal address stack and proceeds to process the children of the operator, which may be other non-terminal or terminal operators, or terminal constants or variables. The operators are represented on the operand stack as map links and the constants or variables exist on the stack as themselves with their repective taggings. The sub-routines PUTX and GENATTR are invoked whenever a terminal constant or variable is encountered.

GENATTR takes items pertaining to variables about to be put onto the execution tree or about to be recorded as keys to entity sets and places the decoded information into the entity set table. The data processed indicate the name of the attribute, any parent of the attribute, and whether the attribute is a key, either virtual information, cartesian, or both.

Hence we have described the operation of the finite-state, push-down automaton parser necessary to encode the user inputs into specifications comprehensible to the entity level for retrieval purposes. The specifications are encoded into pre-formated tables.

6.4.0 SIMPLIFICATION OF REQUEST

The SMPLFY routine minimizes the amount of data flow across the boundary from the virtual information level to the entity level by abridging the entity request table, ENTITY, and creating the retrieval set, RETE_ARG. Only requests for real entity set information exist in RETE_ARG. In addition, the execution tree, XTREE is trimmed by the removal of certain easily manipulated conditions and the placement of these conditions in RETE_ARG for processing in the lower levels of INFOPLEX. Before any of this is done, however, SMPLFY's task is to assign all attributes correctly into the appropriate entity sets, using the propagation function PRPGATE.

PRPGATE sets up the proper relationships between entity sets and attributes, in the sense that attributes used in any entity sets which are composed from other entity sets must exist in the parent entity sets as well. An attribute map table is filled for each composite entity set to make the attribute relationships endure though the retrieval and execution phases of the data processing. All attribute names and parent

relationships are ensured when propagating so as not to create conflicts in naming. For instance, if entity B is based on entity A, then if entity A has attributes X -> Y -> Z (here, we mean that X,Y, and Z are each attributes of A, with X being the parent of Y being the parent of Z) and entity B has attributes X -> U -> Y -> Z then entity B, after propagation, would acquire the new attributes U, Y, and Z, with the parent relationship connected; thus B would have two attributes by the name Y and two by Z.

SEARCH goes through the trees attached to each real entity set and attempts to simplify their structure whenever and only whenever comparisons are made between pairs of variables and constants and which are at most joined by the 'and' joiner. The rationale for processing thus has to do with the power and efficiency of the unary level underlying the VIFI. The trick that SEARCH uses in amputating the tree involves the tagging of nodes so as to only prolong those branches which reach unresolvable nodes in terms of simplification. When a simplification is found, the SETREL sub-routine is invoked to set the relationship of interest into RETE_ARG for computation at the lower levels. An example of a partially resolvable tree would be *(name = 'Bond' or id_num = '007' and job = 'spy')*, where the name and id_num attributes cannot be taken out of the tree but the job attribute can because it is joined to the other conditions by a simple 'and'.

Once simplification is done, the control of the virtual information stage must be temporarily passed down to the entity level for request processing in order for further actions to be performable in the virtual information level. After all, virtualness has to be based upon some form of substance.

6.5.0 CONVERSION OF STORAGE

The CNVERT routine is a simple routine to transfer data from specialized storage necessary to the INFOPLEX control structure to storage more dedicated to the internal environment of the virtual information stage. This transfer allows uniformity of data processing in the VIFI. Of even greater significance is the amount of independence and flexibilty the VIFI achieves from the external environments. Only CNVERT needs to be altered should any interface requirements be updated. Once the information tags and attributes' values of the retrieved real entity sets have found new shelter, CNVERT proceeds to eliminate their old home.

6.6.0 EXECUTION OF DISCRIMINATOR

XECUTE is the meat of the data processing stage of the VIFI. The functions which it performs are creating virtual entity sets and applying additional constraints on completed entity sets for the purpose of discrimination. Here, discrimination is the subsetting of the entity sets based on the user speci-

fied conditioning clauses which were not simplified and sent to the entity level. The sub-routines of XECUTE are divided into two components: the set generation/condensing routines and the subset conditioning routines. They act on the retrieved data which have been attached to the ENTITY query table and base their actions on information in the ENTITY table itself, the execution tree, XTREE, and the additional information table, XCHNGE.

XECUTE has a driver loop that distributes the operations on each entity set in turn. The set generation/condensing routines are:

MUNION which builds a multiple union of two entity sets by copying all the attribute values of the first parent and those values of the second which do not occur in the first.

MINTER which builds a multiple intersection of two entity sets by copying only those attribute values of the first parent that exist in the second.

SINGLE which condenses an entity set by moving all attribute values of the entity set to a dummy entity set and then copying back only those values whose occurrence is the first or only in the entity set.

CRTESN which creates a cartesian set by copying multiple attribute values from each of its parent entity sets. It copies each value of the first parent the number of times dictated by the depth (number of values for a single occurrence attribute) of the second, and copies sets of values of the second parent the number of times dictated by the depth of the first.

The basic sub-routine with which entity sets are copied is COPYSET. It copies an entire set of attribute values from a specified parent entity set to the child, with special care in regards to single or multiple occurrence. Markers are set by the caller to COPYSET to to track the et of attribute values to be copied and these markers are advanced at the end of copying. Level numbers are used to manage the copying of multiple occurrence items. A counter is incremented to track the number of sets of values copied. In the same line is SKIPSET, which skips a set of attribute values by incrementing the abovementioned markers to the specified parent without doing any copying. Hence we can control which sets of attribute values are to be copied to the child. COPYALL, on the other hand, just copies all values from an parent entity to a child.

Of aid to set generation are the sub-routines LOCATEKEYS and IUMATCH. The first sets up a key table to allow easy access to entity keys for the purpose of determining

intersections and unions. The second determines if two sets of attribute values are in fact one, according to the keys assigned.

The subset conditioning routines function recursively to traverse the parser-filled execution tree in order to perform the user specified operations on the attribute values of the real and virtual (composite) entity sets. The argument passing between the recursive calls is through structural duplicates of the linked lists containing the original copies of the values. The main recursive sub-routines are as follows:

OPERATE sequences the processing of the nodes by identifying each node's label and/or number of children, and subsequently distributing the manipulation of the child or children to the other routines, or to a recursive invocation of itself. A node of no child is either a variable, constant, multiple constant, or operator of no argument; the terminal node must therefore create a vertical or unitary list to return to the tree node above. A node of one child is an operator of one argument and hence must return either a unitary or vertical list, depending upon the operator type. A node of two children is an operator with two arguments and may have a multiple constant as a second child. The GO_DOWN routines execute the operations, defined by the labels of the nodes, on their

children, if any exist. The STR routine is a special string operation handler which is also recursive.

STR is a very powerful routine that handles nodes labeled 'str'. It does substring operations on the first child of the 'str' node whith specifications from the other five children. A clearcut description of the 'str' function is in Jameson Lee's description of the front-end of the VIFI. Essentially, STR uses OPERATE on those of the second, third, fifth, and sixth children of the 'str' node that are relevent to the type specification. The fourth child specifies the type of string addressing, absolute or relative, that is desired.

The support routines to the recursive ones above are:

GO_DOWN0 returns a unitary list containing data returned from no argument operators.

GO_DOWN1 checks to see if the operator of one argument that it handles returns unitary lists or vertical lists. If a vertical list is to be returned, it goes through each item in the list to evaluate the return for that item.

RET_UNITARY decides whether a one argument operator is defined to always return a unitary list. It also evaluates such operators with the appropriate operand.

UNARY takes a one argument operator and its argument and computes the return.
MULTEQ2 takes the '=' operator and its two arguments, the second of which must be a multiple constant, and evaluates the results of the first augument values each compared with the multiple list.

GO_DOWN2 determines the type of return that is expected of a generic two argument operator and proceeds to fill out a list with that specification.

BINARY takes a two argument operator and its arguments and computes the return.

CREATE_LIST creates a duplicate list of the values of the specified attribute, to be used for processing in upper nodes.

CREATE_CONST creates a unitary list of the single constant passed to it. It also can create a multiple constant index list which is used to address a multiple constant if the terminal node on which it acts is of that type.

TMPLTE is a function that can be used to do template matching on a string, hence giving the user this additional power in value searching.

DECRSET is used to decrease the number of values in an attribute by moving an entity set to a dummy and copying back only those sets of attribute values that have a corresponding ':true' in the conditioning list.

The routine DISPOSE is useful for ensuring no storage overflow since it is invoked whenever unnecessary storage can be deallocated.

The XECUTE routine that is used to discriminate information to the level specified by the user is thus described.

6.7.0 FINAL PROCESSING OF DATA

Although the DOBY routine has as yet not been implemented, due to unresolved problems regarding output interface, it is envisioned that this routine will essentially use the conditioning sub-routines of the XECUTE routine and return the condition lists to the user. Hence it follows the trend and methodology of our processing. It will return those lists which were specifically mentioned by the user and tag them with the labels, which could be of type virtual information, that should be extracted by the tokenizer in the earlier stages of processing.

The full complement of software for the rear-end is thus described.

7.0.0 DESIGN CONSIDERATIONS AND OVERVIEW

7.1.0 MODULARITY, MODIFIABILITY, AND EFFICIENCY

In terms of modularity, I think we have achieved a pretty good record. Our design strategies concentrate on clean interfacing of routines, with clearcut specifications of interfacing parameters. The horizontal control structure, regulated by the activity coordinators, provide sensitive debugging capabilities not available in vertical programming. The manner of tokenizing and parsing database statements allow for changes in the format of retrieval requests without affecting the inner routines. The way data incoming from the entity level is relocated allows easy changing of interface to the lower level. In general, I have no dissatisfactions concerning these issues.

There is, however, a concern I have regarding the complexity in the manner in which certain manipulations of data are performed. Since a certain amount of efficiency is desired in processing, a lot of not-so-obvious shortcuts were taken to derive results. Hence, documentation is required on the level so extremely clear that it becomes a drudgery. Furthermore, certain already implemented functionalities will not be used simply because not enough clear documentation is available.

The efficiency of the INFOPLEX structure has a lot to be desired. This is but a practical matter of structural design verses kludgy efficiency. In a microprocessor implementation, however, i am sure certain structural concepts would give way to efficiency and yet maintain that its integrity factor.

7.2.0 ADAPTABILITY IN OVERALL INFOPLEX ENVIRONMENT

The current virtual information stage is in concept adaptable to the current INFOPLEX environment, but there are quite a few areas in which conflicts arise because of lack of coordination of effort in the updating of the system as a whole. Even though modularity is desired, certain capabilities cannot be realized unless all stages of the system support those capabilities. With parallel microprocessor processing, the speed consideration in the inherently slow hierarchical control structure would hopefully diminish. The Test Vehicle can only approximate what the actual system can do; it can only illustrate some of the possible ways to view the manner data ought to be managed.

7.3.0 POTENTIAL POWER OF VIRTUAL INFORMATION

I personally can see quite a lot of possibilities in this virtual information idea. Our current implementation is somewhat restrictive in regards to some of the desired features, such as full inferring of facts. It can be adapted to practi-

cally any situation, however. Unfortunately, since the re-implementation of programs in general usually is easier done than the updating of existing ones (I program enough to know this), I fear that our package is only a transient element in the progression of the INFOPLEX system. As it is, we have put into VIFI all that could be allowed in the time alloted, and this already seems far enough advanced in comparison to the other components of the system. Hopefully, sometime all the desired features can be put together and INFOPLEX can see the market.

8.0.0 CONCLUSION

The INFOPLEX project seems to be a promising one. The virtual information concept is quite realizable using various schemes to track the transformation of data. According to the tests we have done to generate entity level requests from user inputs that we would consider fairly unrestrictive in terms of power, the difference between 'virtual' and 'real' information is but a few operations to connect the two. We were able to construct fairly restrictive entity requests when the user entered retrieval commands sometimes more complicated than those found in programming languages, and we were able to resolve the commands to those handled by the lower level, for the sake of efficiency, and those handled at the VIFI, for the sake of computational power and flexibility. The parser's approach to converting user-input commands provides a natural extension to increase the capability of the language. As it is, the user can already define a lot of functions based on the built-in ones. We can easily implement such things as data types or data units by a minimum amount of hacking because we already have a system of user definitions. However, since this is only the prototype of the virtual information processor, we feel that what we have is more than adequate.

The only unfortunate thing is that we were not able to fully test out the processing stages of the VIFI, since we lacked sufficient test data and time to establish the necessary linkages to the other components of INFOPLEX. Hopefully, our VIFI will eventually be integrated into the Test Vehicle, and I am sure it will fit in well in the design.

Bibliography

1. Harry R. Lewis, Christos H. Papadimitriou,
   'Elements of the Theory of Computation'
   Prentice-Hall Software Series

2. David Gries,
   'Compiler Construction for Digital Computers'
   John Wily & Sons

3. Jeffrey Folinus, Stuart E. Madnick, Howard Schutzman,
   'Virtual Information in Data Base Computers'
   Center for Information Systems Research, M.I.T.

4. A Klug, D Tsichritzis,
   Multiple View Support Within the Ansi/Sparc Framework
   Center for Information Systems Research, M.I.T.

5. Meichun Hsu
   'FSTV - The Software Test Vehicle for the Functional
   Hierarchy of the INFOPLEX Data Base Computer'
   Center for Information Systems Research, M.I.T.

6. Thomas A. Standish
   'Data Structure Techniques'
   Addison-Wesley Computer Science Series

7. Aho Ullman
   'Principles of Compiler Design'
   Addison-Wesley Computer Science Series

8. Tak To
   'SHELL: A Simulation for the Software Test Vehicle
   of the INFOPLEX Data Base Computer'
   Center for Information Systems Research, M.I.T.

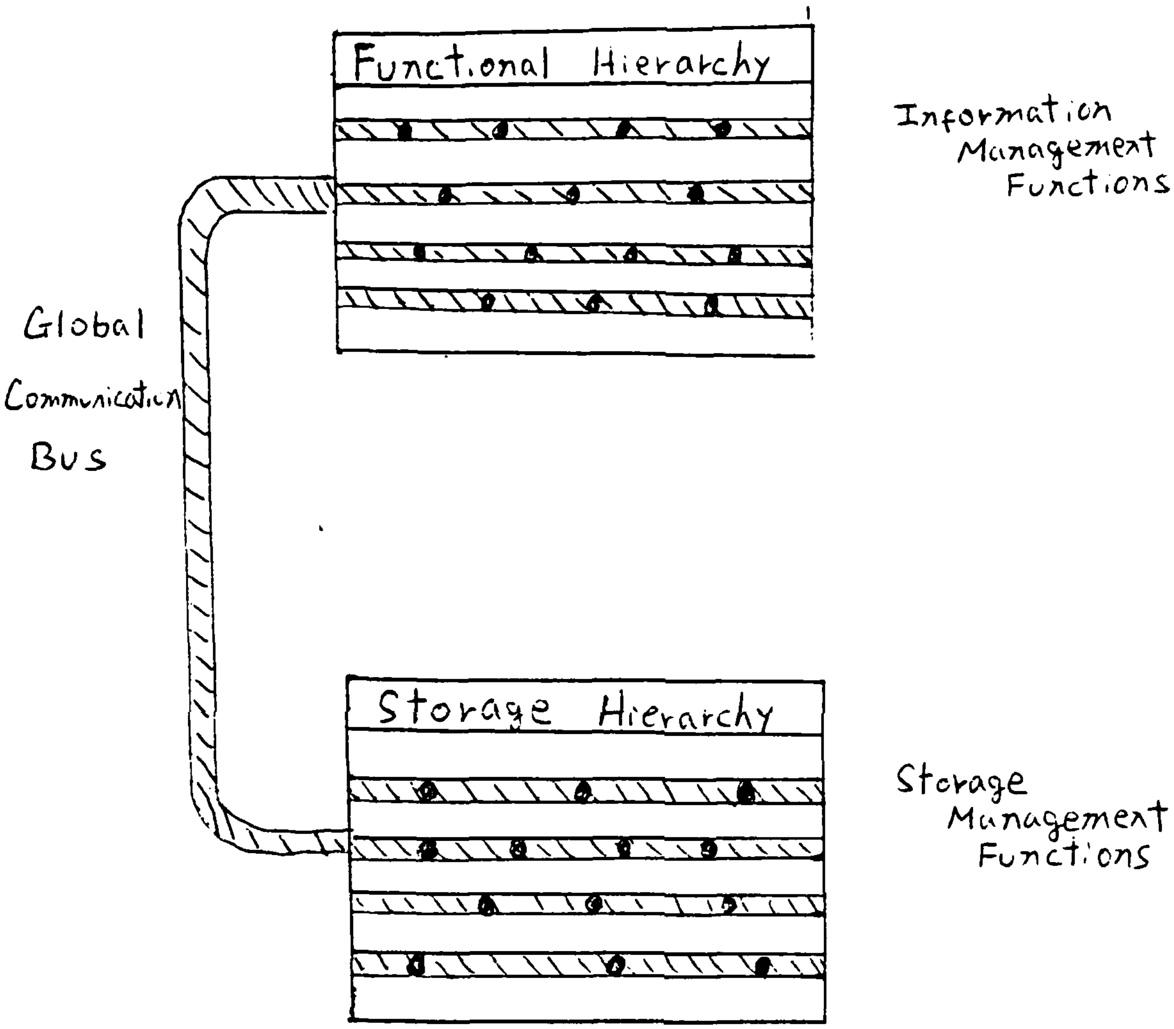
9. Chat-Yu Lam, Stuart E. Madnick
   'INFOPLEX Data Base Computer Architecture'
   Center for Information Systems Research, M.I.T.

10. Bruce Blumberg
    'INFOSAM - A Sample Database Management System'
    Center for Information Systems Research, M.I.T.

Appendix

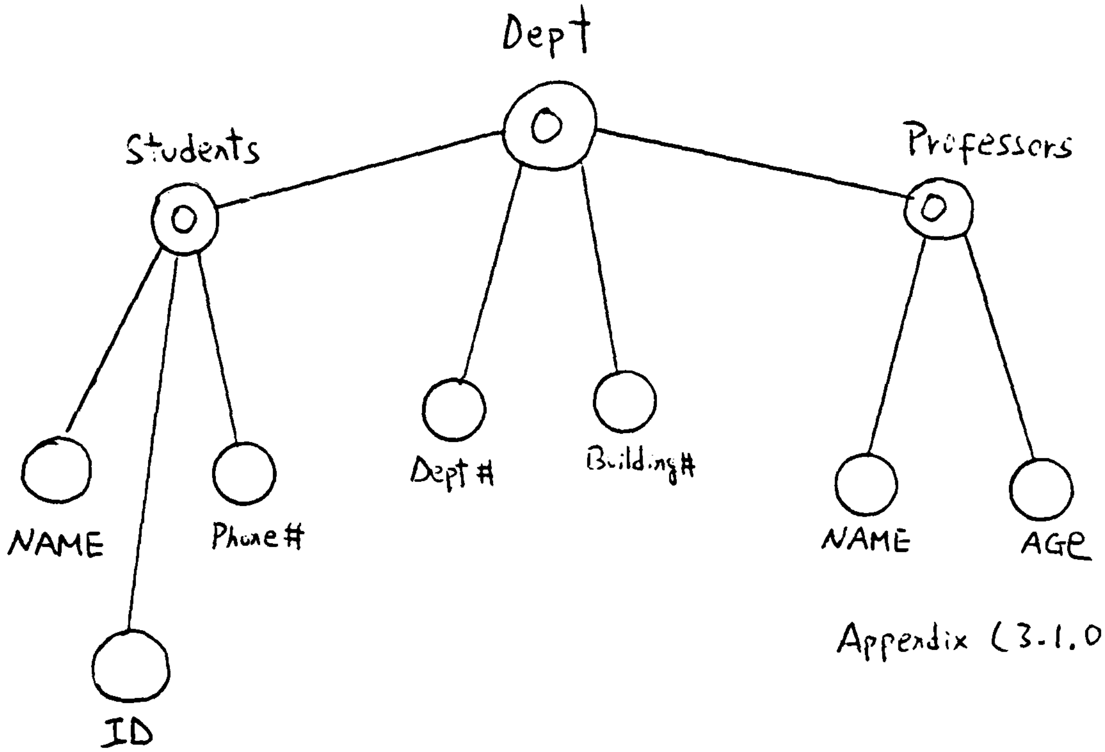
A

INFOPLEX



Appendix A (1.1.2)

Data Model
Data Statements



Appendix (3.1.0)

<u>Database Statements</u>

Define

Adhoc

Listdef

Retrieve

<u>Buffer Commands</u>

Trans	Runtrans
Endtrans	Terminate
Finput	Help
Fsave	Erasebuff
Dodel	Killexec

{Refer to Chapter 5 of Jameson Lee's thesis}

Appendix (3.2.0)

Virtual Definitions

Define heavy as weight > 100 ;

(defined)

Define heavy Remove ; (removed)

Define heavy as weight > 200 ;

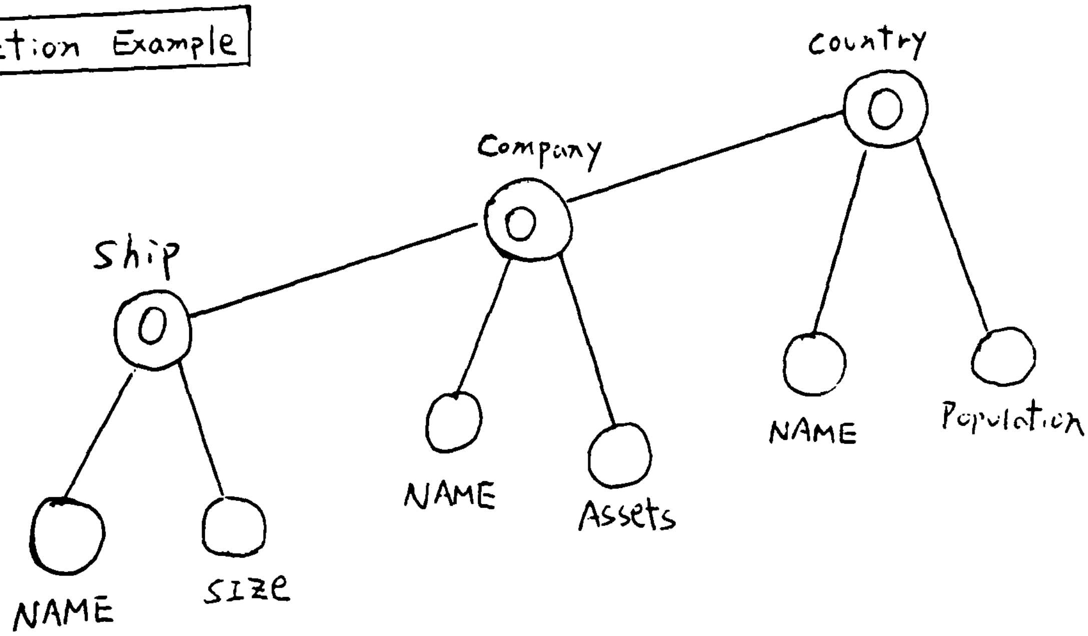
(redefined)

Define heavy as weight > 50 ;

(replaced)

Appendix (3.3.0)

Indirection Example



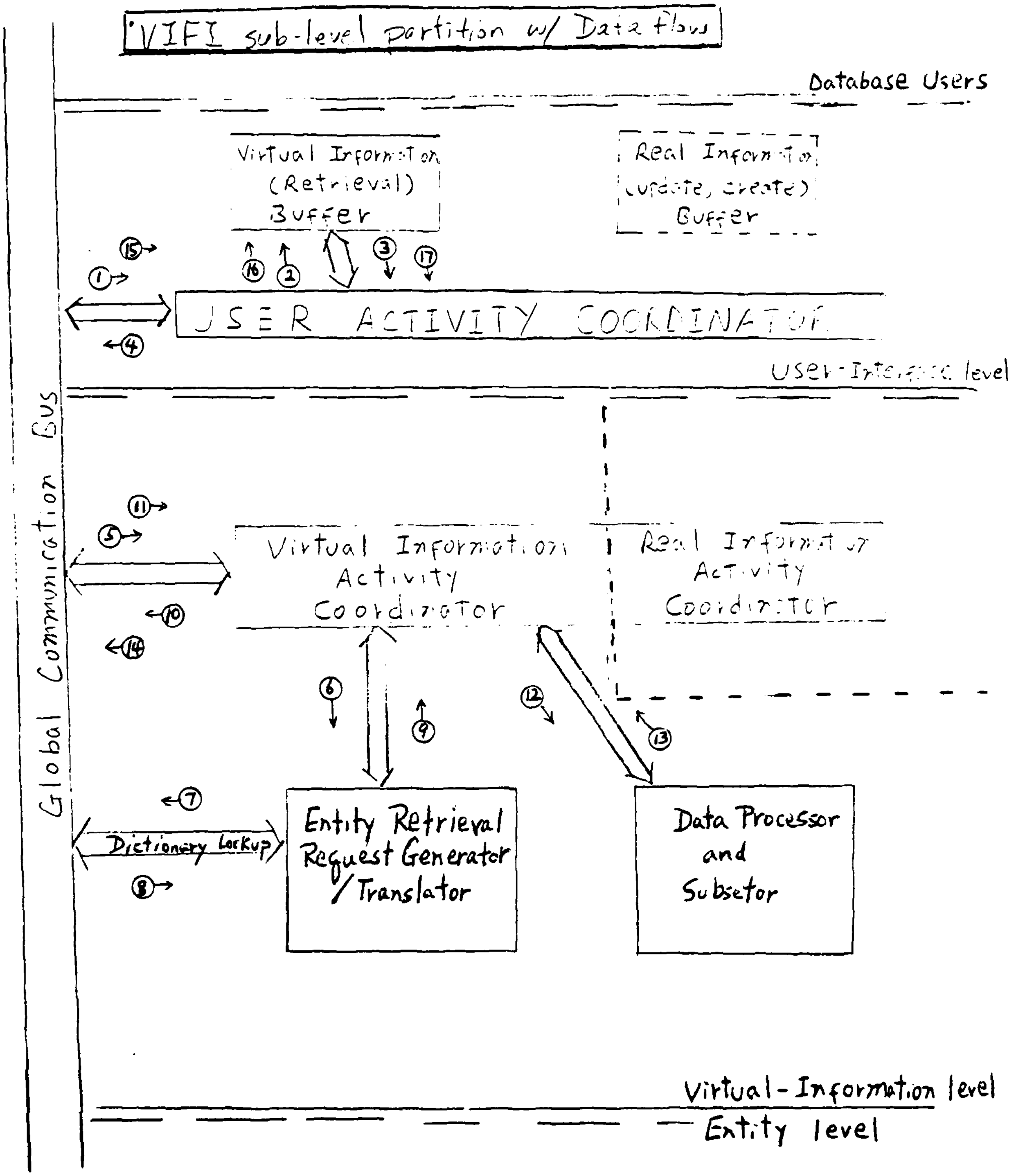
Appendix (3.6.0)

Set Operators

1	MU	multiple union
2	MI	multiple intersection
3	SU	single union
4	SI	single intersection
5	CS	cartesian product

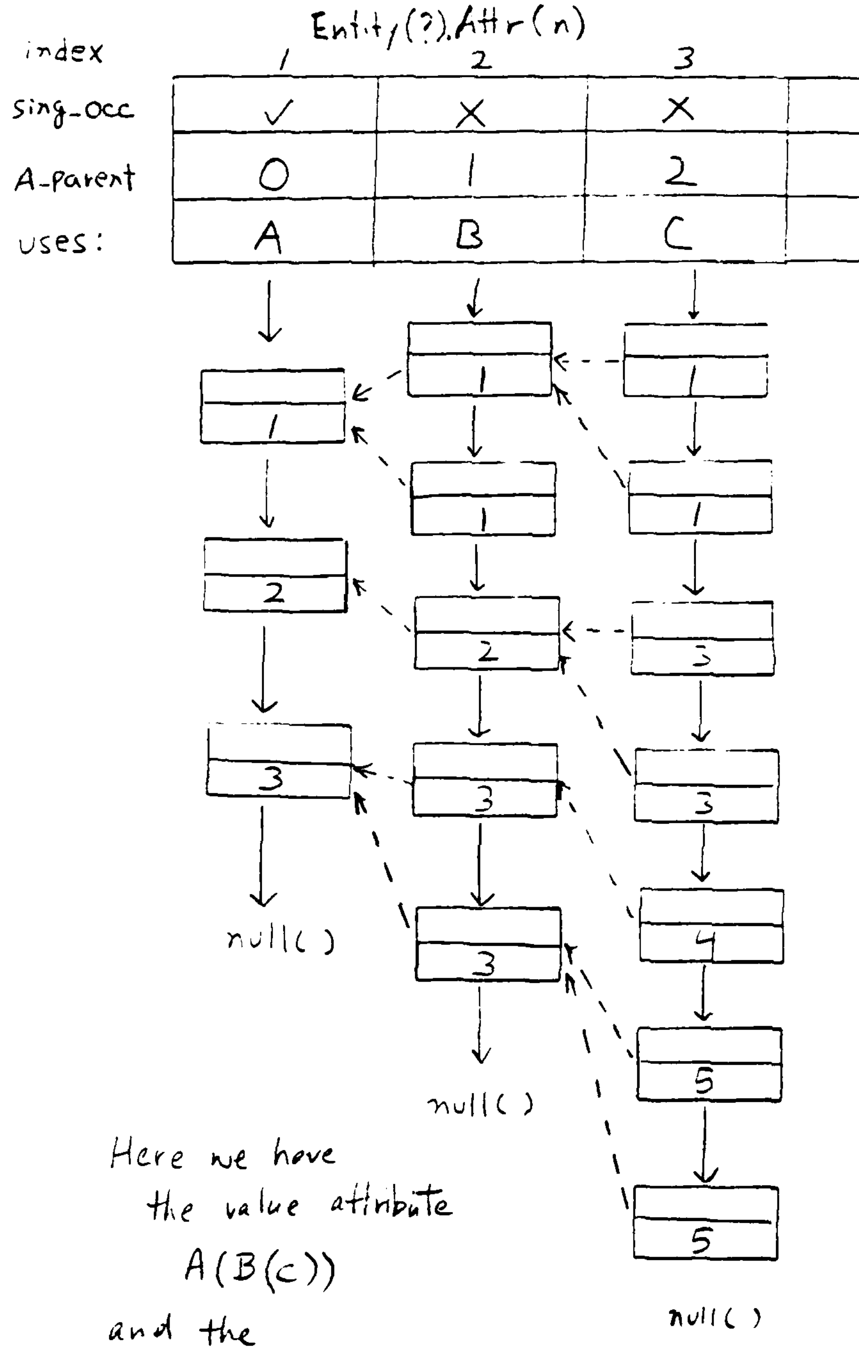
{ Refer to Chapter 5 of Jameson Lee's Thesis }

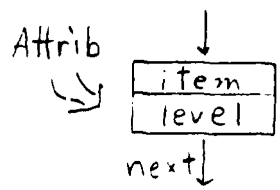
Appendix (3.8.0)



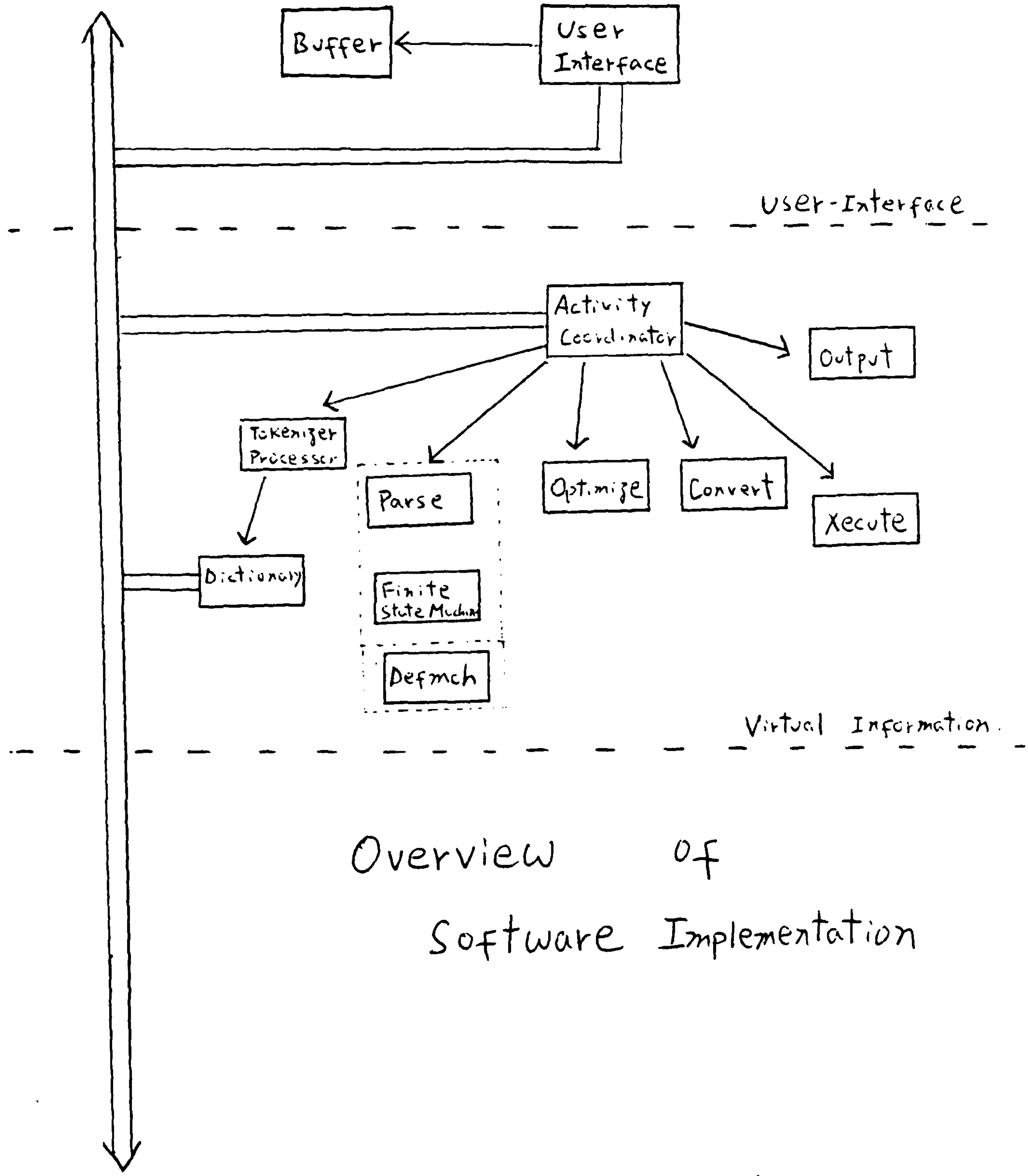
Appendix A (4.1.0)

Attribute Values & Levels





Here we have
the value attribute
A(B(C))
and the
entitity attributes
A(B) and A



Overview of Software Implementation

Appendix (6.0.C)

Appendix

B

```
***********ALL COPY*************                                    ALL00010
                                                                    ALL00020
MACRO           INDEX      SIZE                                     ALL00030
MACH                2         5                                     ALL00040
TOKEN               8         4                                     ALL00050
XTREE              13         4                                     ALL00060
ATTRIB             18         4                                     ALL00070
ENTITY             23        16                                     ALL00080
DICTION            40        13                                     ALL00090
XCHNGE             54         1                                     ALL00100
ARETE              56        32                                     ALL00110
                                                                    ALL00120
```

```
DCL 1 MACH,                                                 ALL00130
      2 STATE_MAP (200) FIXED,                              ALL00140
      2 MATCH (500) CHAR (30) VAR,                          ALL00150
      2 ACTION (500) CHAR (40) VAR,                         ALL00160
      2 NEXT_STATE (500) FIXED;                             ALL00170
                                                            ALL00180
DCL 1 TOKEN BASED,                                          ALL00190
      2 ITEM CHAR (30) VAR,                                 ALL00200
      2 CLASS CHAR (10) VAR,                                ALL00210
      2 NEXT PTR;                                           ALL00220
                                                            ALL00230
DCL 1 XTREE (1000),                                         ALL00240
      2 LABEL CHAR (30) VAR,                                ALL00250
      2 CHILD FIXED,                                        ALL00260
      2 LINK FIXED;                                         ALL00270
                                                            ALL00280
DCL 1 ATTRIB BASED,                                         ALL00290
      2 LEVEL FIXED,                                        ALL00300
      2 ITEM CHAR (50) VAR,                                 ALL00310
      2 NEXT PTR;                                           ALL00320
                                                            ALL00330
DCL 1 ENTITY (0:12),                                        ALL00340
        2 NAME CHAR (30) VAR,                               ALL00350
        2 DEPTH FIXED,                                      ALL00360
        2 VES_FN CHAR (2) VAR,                              ALL00370
        2 WHERE FIXED,                                      ALL00380
        2 N_PARENT (2) FIXED,                               ALL00390
        2 VES_PAR PTR,                                      ALL00400
        2 ATTR (15),                                        ALL00410
           3 VES_KEY BIT (1),                               ALL00420
           3 CART_KEY BIT (1),                              ALL00430
           3 SING_OCC BIT (1),                              ALL00440
           3 A_PARENT FIXED,                                ALL00450
           3 USES CHAR (30) VAR,                            ALL00460
           3 LIST PTR;                                      ALL00470
DCL 1 N_MAP (2) BASED,                                      ALL00480
        2 NUM (15) FIXED;                                   ALL00490
                                                            ALL00500
DCL 1 DICTION,                                              ALL00510
        2 CHECK,                                            ALL00520
           3 LIMIT FIXED,                                   ALL00530
           3 LABEL (50) CHAR (50) VAR,                      ALL00540
           3 TIMES (50) FIXED,                              ALL00550
        2 MAIN,                                             ALL00560
           3 LIMIT FIXED,                                   ALL00570
           3 FROM (100) CHAR (50) VAR,                      ALL00580
           3 TO (100) CHAR (160) VAR,                       ALL00590
        2 ADHOC,                                            ALL00600
           3 LIMIT FIXED,                                   ALL00610
           3 FROM (50) CHAR (50) VAR,                       ALL00620
           3 TO (50) CHAR (160) VAR;                        ALL00630
                                                            ALL00640
DCL XCHNGE (20) CHAR (80) VAR;                              ALL00650
                                                            ALL00660
```

```
COPY ARETE
DCL 1 RETE_ARG BASED (RETEP),  /* WILL ALSO BE RETE_RTN */
      2 CTL_INFO,
        3 LEN FIXED BIN (15),
        3 CBTP FIXED BIN (15) INIT (21),
        3 PTR PTR,
      2 NUM_ATTR FIXED,
      2 NUM_COND FIXED,    /* NUMBER OF CONDITIONS */
      2 ENT,
        3 NAME CHAR (30) VAR,  /* ENTITY SET NAME */
        3 DEPTH FIXED,     /* FILLED IN WHEN RETURNED */
        3 ATTR (NATTR REFER (RETE_ARG.NUM_ATTR)),
          4 SING_OCC BIT (1), /* IF SINGLE OCCUR THEN SET */
          4 A_PARENT FIXED, /* PARENT NUMBER */
          4 USES CHAR(30) VAR, /* ATTRIBUTE NAME */
          4 LIST PTR, /* POINT TO LIST OF OCC OF THIS ATTR IF ANY */
      2 COND (NCOND REFER (RETE_ARG.NUM_COND)),
        3 ATTRREF FIXED, /* POINT TO ATTR IN ATTR ARRAY ABOVE */
        3 NEG BIT(1), /* NEGATION OF RELATOR */
        3 REL CHAR(1), /* '=', '<', '>' */
        3 CDATA (10) CHAR (50) VAR,  /* UP TO 10 'MULTI' ITMES */
      2 RTN_CODE FIXED BIN,
      2 NEXT_PTR PTR;
DCL RETEP PTR;

DCL 1 RETE_RTN1 BASED,           /* USED WHEN RETURNED */
      2 CTL,
        3 LEN FIXED BIN(31),
        3 CBTP FIXED BIN (31) INIT (46),
        3 PTR PTR,
      2 LEVEL FIXED BIN, /* OCCUR NUMBER */
      2 ITEM CHAR (50),
      2 NEXT_PTR PTR; /* POINT TO NEXT ON THE SAME ATTRIBUTE LIST */
```

71

```
 NUMBER LEV NT

     10        0  DCTNRY: PROC (DICTION, COMMSTR) RETURNS (CHAR (160) VAR);                DCT00010
                                                                                           DCT00020
                  %INCLUDE DICTION;**************************************************DCT00030
 100010    1   0  DCL 1 DICTION,                                                           DIC00010
                         2 CHECK,                                                          DIC00020
                            3 LIMIT FIXED,                                                 DIC00030
                            3 LABEL (50) CHAR (50) VAR,                                    DIC00040
                            3 TIMES (50) FIXED,                                            DIC00050
                         2 MAIN,                                                           DIC00060
                            3 LIMIT FIXED,                                                 DIC00070
                            3 FROM (100) CHAR (50) VAR,                                    DIC00080
                            3 TO (100) CHAR (160) VAR,                                     DIC00090
                         2 ADHOC,                                                          DIC00100
                            3 LIMIT FIXED,                                                 DIC00110
                            3 FROM (50) CHAR (50) VAR,                                     DIC00120
                            3 TO (50) CHAR (160) VAR;                                      DIC00130
                  *****************                                                        DCT00030
 200040    1   0  DCL (COMMSTR,FINDSTR) CHAR (160) VAR,                                    DCT00040
                      COMMAND CHAR (50) VAR;                                               DCT00050
                                                                                           DCT00060
 200070    1   0  COMMAND = GETS (COMMSTR,',');                                            DCT00070
 200080    1   0  SELECT (COMMAND);                                                        DCT00080
 200090    1   1     WHEN ('FIND') DO;                                                     DCT00090
 200100    1   2        IF BIF_IDENT (COMMSTR) THEN                                        DCT00100
                           IF INDEX ('(,),{,},;,%',SUBSTR (COMMSTR,1,1)) = 0 THEN          DCT00110
                              RETURN (COMMSTR || ',:B');                                   DCT00120
 200130    1   2           ELSE                                                            DCT00130
                              RETURN (COMMSTR || ',:D');                                   DCT00140
 200150    1   2        FINDSTR = FIND (DICTION.ADHOC);                                    DCT00150
 200160    1   2        IF FINDSTR ^= 'NOT_FOUND' THEN                                     DCT00160
                           RETURN (FINDSTR);                                               DCT00170
 200180    1   2        FINDSTR = FIND (DICTION.MAIN);                                     DCT00180
 200190    1   2        IF FINDSTR ^= 'NOT_FOUND' THEN                                     DCT00190
                           RETURN (FINDSTR);                                               DCT00200
 200210    1   2        RETURN (COMMSTR || ',:R');                                         DCT00210
 200220    1   2        END;                                                               DCT00220
 200230    1   1     WHEN ('SAVE')                                                         DCT00230
                        RETURN (SAVE (DICTION.MAIN,'MAIN'));                               DCT00240
 200250    1   1     WHEN ('ADHSAVE')                                                      DCT00250
                        RETURN (SAVE (DICTION.ADHOC,'ADHOC'));                             DCT00260
 200270    1   1     WHEN ('REMOVE')                                                       DCT00270
                        RETURN (REMOVE (DICTION.MAIN,'MAIN'));                             DCT00280
 200290    1   1     WHEN ('ADHREMOVE')                                                    DCT00290
                        RETURN (REMOVE (DICTION.ADHOC,'ADHOC'));                           DCT00300
 200310    1   1     WHEN ('INITIALIZE') DO;                                               DCT00310
 200320    1   2        DICTION.MAIN.LIMIT = 0;                                            DCT00320
 200330    1   2        DICTION.MAIN.FROM (*) = '';                                        DCT00330
 200340    1   2        DICTION.MAIN.TO (*) = '';                                          DCT00340
 200350    1   2        RETURN ('DICTION_MAIN_INITIALIZED');                               DCT00350
```

```
 NUMBER LEV NT

 200360   1  2        END;                                                         DCT00360
 200370   1  1     WHEN ('ADHCNTRCLR') DO;                                         DCT00370
 200380   1  2        DICTION.ADHOC.LIMIT = 0;                                     DCT00380
 200390   1  2        DICTION.ADHOC.FROM (*) = '';                                 DCT00390
 200400   1  2        DICTION.ADHOC.TO (*) = '';                                   DCT00400
 200410   1  2        DICTION.CHECK.LIMIT = 0;                                     DCT00410
 200420   1  2        DICTION.CHECK.LABEL (*) = '';                                DCT00420
 200430   1  2        DICTION.TIMES (*) = 0;                                       DCT00430
 200440   1  2        RETURN ('DICTION_ADHOC_AND_COUNTER_CLEARED');                DCT00440
 200450   1  2        END;                                                         DCT00450
 200460   1  1     OTHERWISE                                                       DCT00460
                      RETURN ('COMMAND_NOT_FOUND,' || COMMAND);                    DCT00470
 200480   1  1     END;                                                            DCT00480
                                                                                   DCT00490
                                                                                   DCT00500
```

```
NUMBER LEV NT

200510   1  0  BIF_IDENT: PROC (IDENT_ITEM) RETURNS (BIT (1));

200530   2  0  DCL IDENT_ITEM CHAR (*) VAR,
                   BIF_ITEMS CHAR (200) VAR,
                   BIF_ITEM CHAR (50) VAR;

200570   2  0  BIF_ITEMS = '(.),(.),;,+,-,*,/,¦,!,>,<,=,^,AND,OR,XOR,DATE,'
                         || 'ZER,POS,SGN,ABS,DEPTH,MAX,MIN,SUM,STR,'
                         || 'RETRIEVE,BY,SORT,MU,SU,MI,SI,CS,'
                         || '%0,%1,%2,%3,%4,%4,%5,%6,%7,%8,%9';
200610   2  0  BIF_ITEM = GETS (BIF_ITEMS,',');
200620   2  0  DO WHILE (BIF_ITEMS ^= '');
200630   2  1     IF BIF_ITEM = IDENT_ITEM THEN
                     RETURN ('1'B);
200650   2  1     BIF_ITEM = GETS (BIF_ITEMS,',');
200660   2  1     END;
200670   2  0  RETURN ('0'B);

200690   2  0  END BIF_IDENT;
```

```
NUMBER LEV NT

200720   1  0  FIND: PROC (SUBDICTION) RETURNS (CHAR (160) VAR);                     DCT00720
                                                                                      DCT00730
200740   2  0  DCL 1 SUBDICTION,                                                      DCT00740
                     2 LIMIT FIXED,                                                   DCT00750
                     2 FROM (*) CHAR (50) VAR,                                        DCT00760
                     2 TO (*) CHAR (160) VAR;                                         DCT00770
200780   2  0  DCL (LOOKSTR,MATCHSTR,REPSTR) CHAR (160) VAR;                          DCT00780
                                                                                      DCT00790
200800   2  0  LOOKSTR = COMMSTR;                                                     DCT00800
200810   2  0  IF SUBDICTION.LIMIT > 0 THEN                                           DCT00810
                  DO I = 1 TO SUBDICTION.LIMIT;                                       DCT00820
200830   2  1       MATCHSTR = SUBDICTION.FROM (I);                                   DCT00830
200840   2  1       IF GETS (LOOKSTR,'#') = GETS (MATCHSTR,'#') THEN DO;              DCT00840
200850   2  2          REPSTR = SUBDICTION.TO (I);                                    DCT00850
200860   2  2          IF OK_TIMES THEN DO;                                           DCT00860
200870   2  3             DO WHILE (MATCHSTR ^= '');                                  DCT00870
200880   2  4                CALL REPLACE (REPSTR,                                    DCT00880
                                   '#' || GETS (MATCHSTR,'#'),                        DCT00890
                                   GETS (LOOKSTR,'#'));                               DCT00900
200910   2  4                END;                                                     DCT00910
200920   2  3             RETURN (REPSTR || ',:V');                                   DCT00920
200930   2  3             END;                                                        DCT00930
200940   2  2          ELSE                                                           DCT00940
                          RETURN ('1*ERROR* TOO_MANY_FIND_DEFINITIONS,'               DCT00950
                                  || COMMSTR);                                        DCT00960
200970   2  2          END;                                                           DCT00970
200980   2  1       END;                                                              DCT00980
200990   2  0  RETURN ('NOT_FOUND');                                                  DCT00990
                                                                                      DCT01000
201010   2  0  END FIND;                                                              DCT01010
                                                                                      DCT01020
                                                                                      DCT01030
```

```
NUMBER LEV NT

201040   1  0  OK_TIMES: PROC RETURNS (BIT (1));                                DCT01040
                                                                                DCT01050
201060   2  0  DCL (I,J) FIXED;                                                 DCT01060
                                                                                DCT01070
201080   2  0  IF DICTION.CHECK.LIMIT > 0 THEN                                  DCT01080
                  DO I = 1 TO DICTION.CHECK.LIMIT;                              DCT01090
201100   2  1        IF DICTION.CHECK.LABEL (I) = COMMSTR THEN DO;             DCT01100
201110   2  2           J = I;                                                  DCT01110
201120   2  2           I = 200;                                                DCT01120
201130   2  2           END;                                                    DCT01130
201140   2  1        END;                                                       DCT01140
201150   2  0  ELSE                                                             DCT01150
                  I = 0;                                                        DCT01160
201170   2  0  IF I < 200 THEN DO;                                              DCT01170
201180   2  1     DICTION.CHECK.LIMIT = DICTION.CHECK.LIMIT + 1;                DCT01180
201190   2  1     J = DICTION.CHECK.LIMIT;                                      DCT01190
201200   2  1     DICTION.CHECK.LABEL (J) = COMMSTR;                            DCT01200
201210   2  1     END;                                                          DCT01210
201220   2  0  DICTION.CHECK.TIMES (J) = DICTION.CHECK.TIMES (J) + 1;           DCT01220
201230   2  0  RETURN (DICTION.CHECK.TIMES (J) <= 10);                          DCT01230
                                                                                DCT01240
201250   2  0  END OK_TIMES;                                                    DCT01250
                                                                                DCT01260
                                                                                DCT01270
```

```
NUMBER LEV NT

201280   1  0  SAVE: PROC (SUBDICTION,DICTNAME) RETURNS (CHAR (160) VAR);         DCTO1280
                                                                                  DCTO1290
201300   2  0  DCL 1 SUBDICTION,                                                  DCTO1300
                     2 LIMIT FIXED,                                               DCTO1310
                     2 FROM (*) CHAR (50) VAR,                                    DCTO1320
                     2 TO (*) CHAR (160) VAR;                                     DCTO1330
201340   2  0  DCL DICTNAME CHAR (5) VAR,                                         DCTO1340
                   LOOKUP CHAR (50) VAR,                                          DCTO1350
                   (FIRSTSP,I) FIXED;                                             DCTO1360
                                                                                  DCTO1370
201380   2  0  LOOKUP = GETS (COMMSTR,',');                                       DCTO1380
201390   2  0  IF BIF_IDENT (LOOKUP) THEN                                         DCTO1390
                  RETURN ('*ERROR* ATTEMPT_TO_REDEFINE_BIF,'                      DCTO1400
                          || LOOKUP);                                             DCTO1410
201420   2  0  FIRSTSP = 0;                                                       DCTO1420
201430   2  0  IF SUBDICTION.LIMIT > 0 THEN                                       DCTO1430
                  DO I = 1 TO SUBDICTION.LIMIT;                                   DCTO1440
201450   2  1        IF SUBDICTION.FROM (I) = LOOKUP THEN DO;                     DCTO1450
201460   2  2           SUBDICTION.TO (I) = COMMSTR;                              DCTO1460
201470   2  2           RETURN ('REPLACED_IN_' || DICTNAME || ',' || LOOKUP);     DCTO1470
201480   2  2           END;                                                      DCTO1480
201490   2  1        ELSE IF SUBDICTION.FROM (I) = '' & FIRSTSP = 0 THEN          DCTO1490
                        FIRSTSP = I;                                              DCTO1500
201510   2  1        END;                                                         DCTO1510
201520   2  0  IF FIRSTSP = 0 THEN DO;                                            DCTO1520
201530   2  1     SUBDICTION.LIMIT = SUBDICTION.LIMIT + 1;                        DCTO1530
201540   2  1     FIRSTSP = SUBDICTION.LIMIT;                                     DCTO1540
201550   2  1     END;                                                            DCTO1550
201560   2  0  SUBDICTION.FROM (FIRSTSP) = LOOKUP;                                DCTO1560
201570   2  0  SUBDICTION.TO (FIRSTSP) = COMMSTR;                                 DCTO1570
201580   2  0  RETURN ('SAVED_IN_' || DICTNAME || ',' || LOOKUP);                 DCTO1580
                                                                                  DCTO1590
201600   2  0  END SAVE;                                                          DCTO1600
                                                                                  DCTO1610
                                                                                  DCTO1620
```

```
NUMBER LEV NT

201630   1  0  REMOVE. PROC (SUBDICTION,DICTNAME) RETURNS (CHAR (160) VAR);

201650   2  0  DCL 1 SUBDICTION,
                     2 LIMIT FIXED,
                     2 FROM (*) CHAR (50) VAR,
                     2 TO (*) CHAR (160) VAR;
201690   2  0  DCL DICTNAME CHAR (5) VAR,
                   I FIXED;

201720   2  0  IF BIF_IDENT (COMMSTR) THEN
                  RETURN ('*ERROR* ATTEMPT_TO_REMOVE_BIF,' || COMMSTR);
201740   2  0  IF SUBDICTION.LIMIT > 0 THEN
                  DO I = 1 TO SUBDICTION.LIMIT;
201760   2  1        IF SUBDICTION.FROM (I) = COMMSTR THEN DO;
201770   2  2           SUBDICTION.FROM (I) = '';
201780   2  2           SUBDICTION.TO (I) = '';
201790   2  2           RETURN ('REMOVED_FROM_' || DICTNAME || ',' || COMMSTR);
201800   2  2           END;
201810   2  1        END;
201820   2  0  RETURN ('*ERROR* NOT_FOUND_IN_' || DICTNAME || ',' || COMMSTR);

201840   2  0  END REMOVE;
```

```
NUMBER LEV NT

201870   1  0  REPLACE: PROC (RSTR,FROMSTR,TOSTR);                                  DCT01870
                                                                                    DCT01880
201890   2  0  DCL (RSTR,REPLSTR,FROMSTR,TOSTR) CHAR (160) VAR,                     DCT01890
                    I FIXED;                                                        DCT01900
                                                                                    DCT01910
201920   2  0  REPLSTR = '';                                                        DCT01920
201930   2  0  I = INDEX (RSTR,FROMSTR);                                            DCT01930
201940   2  0  DO WHILE (I ^= 0);                                                   DCT01940
201950   2  1     REPLSTR= REPLSTR || SUBSTR (RSTR,1,I - 1) || TOSTR;               DCT01950
201960   2  1     RSTR = SUBSTR (RSTR,I + LENGTH (FROMSTR));                        DCT01960
201970   2  1     I = INDEX (RSTR,FROMSTR);                                         DCT01970
201980   2  1     END;                                                              DCT01980
201990   2  0  RSTR = REPLSTR || RSTR;                                              DCT01990
                                                                                    DCT02000
202010   2  0  END REPLACE;                                                         DCT02010
                                                                                    DCT02020
                                                                                    DCT02030
```

```
NUMBER LEV NT

202040   1  0  GETS: PROC (LIST,TERM_ITEM) RETURNS (CHAR (160) VAR);          DCTO2040
                                                                                DCTO2050
202060   2  0  DCL LIST CHAR (*) VAR,                                          DCTO2060
                   TERM_ITEM CHAR (1),                                         DCTO2070
                   RTN_LIST CHAR (160) VAR,                                    DCTO2080
                   I FIXED;                                                    DCTO2090
                                                                                DCTO2100
202110   2  0  I = INDEX (LIST,TERM_ITEM);                                     DCTO2110
202120   2  0  IF I = 0 THEN DO;                                               DCTO2120
202130   2  1     RTN_LIST = LIST;                                             DCTO2130
202140   2  1     LIST = '';                                                   DCTO2140
202150   2  1     END;                                                         DCTO2150
202160   2  0  ELSE DO;                                                        DCTO2160
202170   2  1     RTN_LIST = SUBSTR (LIST,1,I - 1);                            DCTO2170
202180   2  1     LIST = SUBSTR (LIST,I + 1);                                  DCTO2180
202190   2  1     END;                                                         DCTO2190
202200   2  0  RETURN (RTN_LIST);                                              DCTO2200
                                                                                DCTO2210
202220   2  0  END GETS;                                                       DCTO2220
                                                                                DCTO2230
                                                                                DCTO2240
202250   1  0  END DCTNRY;                                                     DCTO2250
                                                                                DCTO2260
                                                                                DCTO2270
                                                                                DCTO2280
                                                                                DCTO2290
                                                                                DCTO2300
```

```
 NUMBER LEV NT

     10       0  DEFMCH: PROC (MACH);                                                      DEF00010
                                                                                           DEF00020
                 %INCLUDE MACH;***********************************************************DEF00030
 100010   1   0  DCL 1 MACH,                                                               MAC00010
                     2 STATE_MAP (200) FIXED,                                              MAC00020
                     2 MATCH (500) CHAR (30) VAR,                                          MAC00030
                     2 ACTION (500) CHAR (40) VAR,                                         MAC00040
                     2 NEXT_STATE (500) FIXED;                                             MAC00050
                 **************                                                            DEF00030
 200040   1   0  DCL MACHIN FILE RECORD;                                                   DEF00040
 200050   1   0  DCL FREE_LOC FIXED INIT (1);                                              DEF00050
 200060   1   0  DCL LINE CHAR (80),                                                       DEF00060
                     (LINE1,TYPE) CHAR (80) VAR;                                           DEF00070
                                                                                           DEF00080
 200090   1   0  ON ENDFILE (MACHIN) GOTO COMPLETE;                                        DEF00090
                                                                                           DEF00100
 200110   1   0  OPEN FILE (MACHIN) INPUT;                                                 DEF00110
 200120   1   0  DO WHILE ('1'B);                                                          DEF00120
 200130   1   1     READ FILE (MACHIN) INTO (LINE);                                        DEF00130
 200140   1   1     LINE1 = LINE;                                                          DEF00140
 200150   1   1     TYPE = GETS (LINE1,'\');                                               DEF00150
 200160   1   1     IF TYPE = 'S' THEN                                                     DEF00160
                       MACH.STATE_MAP (FIXED (GETS (LINE1,'\'))) = FREE_LOC;               DEF00170
 200180   1   1     ELSE IF TYPE = 'T' THEN DO;                                            DEF00180
 200190   1   2        MACH.MATCH (FREE_LOC) = GETS (LINE1,'\');                           DEF00190
 200200   1   2        MACH.ACTION (FREE_LOC) = GETS (LINE1,'\');                          DEF00200
 200210   1   2        MACH.NEXT_STATE (FREE_LOC) = FIXED (GETS (LINE1,'\'));              DEF00210
 200220   1   2        FREE_LOC = FREE_LOC + 1;                                            DEF00220
 200230   1   2        END;                                                                DEF00230
 200240   1   1     END;                                                                   DEF00240
                                                                                           DEF00250
```

```
NUMBER LEV NT

200260   1  0  GETS: PROC (LIST,TERM_ITEM) RETURNS (CHAR (80) VAR);          DEF00260
                                                                            DEF00270
200280   2  0  DCL LIST CHAR (*) VAR,                                        DEF00280
                   TERM_ITEM CHAR (1),                                       DEF00290
                   RTN_LIST CHAR (80) VAR,                                   DEF00300
                   I FIXED;                                                  DEF00310
                                                                            DEF00320
200330   2  0  I = INDEX (LIST,TERM_ITEM);                                   DEF00330
200340   2  0  IF I = 0 THEN DO;                                             DEF00340
200350   2  1     RTN_LIST = LIST;                                           DEF00350
200360   2  1     LIST = '';                                                 DEF00360
200370   2  1     END;                                                       DEF00370
200380   2  0  ELSE DO;                                                      DEF00380
200390   2  1     RTN_LIST = SUBSTR (LIST,1,I - 1);                          DEF00390
200400   2  1     LIST = SUBSTR (LIST,I + 1);                                DEF00400
200410   2  1     END;                                                       DEF00410
200420   2  0  RETURN (RTN_LIST);                                            DEF00420
                                                                            DEF00430
200440   2  0  END GETS;                                                     DEF00440
                                                                            DEF00450
200460   1  0  COMPLETE:                                                     DEF00460
               PUT SKIP LIST ('MACHINE DEFINITION COMPLETE');                DEF00470
200480   1  0  CLOSE FILE (MACHIN);                                          DEF00480
                                                                            DEF00490
```

```
NUMBER LEV NT
200500   1  0  END DEFMCH;                                    DEF00500
```

```
NUMBER LEV NT

    10       0  PARSE: PROC (MACH,TOKENS_PTR,XTREE,XCHNGE,ENTITY,DEBUG);                  PAR00010
                                                                                          PAR00020
                %INCLUDE MACH;***************************************************************PAR00030
100010   1   0  DCL 1 MACH,                                                               MAC00010
                      2 STATE_MAP (200) FIXED,                                            MAC00020
                      2 MATCH (500) CHAR (30) VAR,                                        MAC00030
                      2 ACTION (500) CHAR (40) VAR,                                       MAC00040
                      2 NEXT_STATE (500) FIXED;                                           MAC00050
                *************                                                             PAR00030
                %INCLUDE TOKEN;**************************************************************PAR00040
300010   1   0  DCL 1 TOKEN BASED,                                                        TOK00010
                      2 ITEM CHAR (30) VAR,                                               TOK00020
                      2 CLASS CHAR (10) VAR,                                              TOK00030
                      2 NEXT PTR;                                                         TOK00040
                **************                                                            PAR00040
                %INCLUDE XTREE;**************************************************************PAR00050
500010   1   0  DCL 1 XTREE (1000),                                                       XTR00010
                      2 LABEL CHAR (30) VAR,                                              XTR00020
                      2 CHILD FIXED,                                                      XTR00030
                      2 LINK FIXED;                                                       XTR00040
                **************                                                            PAR00050
                %INCLUDE XCHNGE;*************************************************************PAR00060
600010   1   0  DCL XCHNGE (20) CHAR (80) VAR;                                            IND00010
                ***************                                                           PAR00060
                %INCLUDE ENTITY;*************************************************************PAR00070
700010   1   0  DCL 1 ENTITY (0:12),                                                      ENT00010
                         2 NAME CHAR (30) VAR,                                            ENT00020
                         2 DEPTH FIXED,                                                   ENT00030
                         2 VES_FN CHAR (2) VAR,                                           ENT00040
                         2 WHERE FIXED,                                                   ENT00050
                         2 N_PARENT (2) FIXED,                                            ENT00060
                         2 VES_PAR PTR,                                                   ENT00070
                         2 ATTR (15),                                                     ENT00080
                            3 VES_KEY BIT (1),                                            ENT00090
                            3 CART_KEY BIT (1),                                           ENT00100
                            3 SING_OCC BIT (1),                                           ENT00110
                            3 A_PARENT FIXED,                                             ENT00120
                            3 USES CHAR (30) VAR,                                         ENT00130
                            3 LIST PTR;                                                   ENT00140
700150   1   0  DCL 1 N_MAP (2) BASED,                                                    ENT00150
                         2 NUM (15) FIXED;                                                ENT00160
                ***************                                                           PAR00070
                                                                                          PAR00080
800090   1   0  DCL TOKENS_PTR PTR;                                                       PAR00090
800100   1   0  DCL DEBUG BIT (1);                                                        PAR00100
800110   1   0  DCL (XTREE_LOC,XCH_LOC,CUR_ES) FIXED;                                     PAR00110
800120   1   0  DCL STACK (3,0:100) CHAR (30) VAR,                                        PAR00120
                    TOS (3) FIXED;                                                        PAR00130
800140   1   0  DCL VES (0:9) FIXED,                                                      PAR00140
```

```
NUMBER LEV NT

                   VES_LOC FIXED;                                  PAR00150
800160   1  0  DCL (STATE_NUM,N) FIXED,                            PAR00160
                   (INPUT_PTR,FREE_PTR) PTR;                       PAR00170
800180   1  0  DCL ERROR FILE OUTPUT;                              PAR00180
800190   1  0  OPEN FILE (ERROR);                                  PAR00190
                                                                   PAR00200
800210   1  0  XTREE (*).LABEL = '';                               PAR00210
800220   1  0  XTREE (*).CHILD = 0;                                PAR00220
800230   1  0  XTREE (*).LINK = 0;                                 PAR00230
800240   1  0  XTREE_LOC = 1;                                      PAR00240
800250   1  0  XCH_LOC = 0;                                        PAR00250
800260   1  0  ENTITY (*).NAME = '';                               PAR00260
800270   1  0  ENTITY (*).DEPTH = 0;                               PAR00270
800280   1  0  ENTITY (*).VES_FN = '';                             PAR00280
800290   1  0  ENTITY (*).WHERE = 0;                               PAR00290
800300   1  0  ENTITY (*).N_PARENT (*) = 0;                        PAR00300
800310   1  0  ENTITY (*).VES_PAR = NULL ();                       PAR00310
800320   1  0  ENTITY (*).ATTR (*).VES_KEY = '0'B;                 PAR00320
800330   1  0  ENTITY (*).ATTR (*).CART_KEY = '0'B;                PAR00330
800340   1  0  ENTITY (*).ATTR (*).SING_OCC = '1'B;                PAR00340
800350   1  0  ENTITY (*).ATTR (*).A_PARENT = 0;                   PAR00350
800360   1  0  ENTITY (*).ATTR (*).USES = '';                      PAR00360
800370   1  0  ENTITY (*).ATTR (*).LIST = NULL ();                 PAR00370
800380   1  0  CUR_ES = 0;                                         PAR00380
800390   1  0  STACK (*,0) = '@BOS';                               PAR00390
800400   1  0  TOS (*) = 0;                                        PAR00400
800410   1  0  VES (*) = 0;                                        PAR00410
800420   1  0  VES_LOC = -1;                                       PAR00420
800430   1  0  INPUT_PTR = TOKENS_PTR;                             PAR00430
800440   1  0  STATE_NUM = 1;                                      PAR00440
                                                                   PAR00450
```

```
NUMBER LEV NT

800460   1  0  DO WHILE (STATE_NUM ^= 0);                                         PAR00460
800470   1  1     IF DEBUG THEN                                                   PAR00470
                     PUT SKIP (2) EDIT                                            PAR00480
                        ('TRANSIT: STATE = ',STATE_NUM,                           PAR00490
                         'INPUT =$',INPUT_PTR -> TOKEN.ITEM,'$$',                 PAR00500
                         'CLASS =$',INPUT_PTR -> TOKEN.CLASS,'$$',                PAR00510
                         'STK#1 =$', STACK (1,TOS (1)),'$$',                      PAR00520
                         'STK#2 =$', STACK (2,TOS (2)),'$$')                      PAR00530
                        (A,F(5),4(SKIP,A,A,A));                                   PAR00540
800550   1  1     DO N = MACH.STATE_MAP (STATE_NUM)                               PAR00550
                         TO MACH.STATE_MAP (STATE_NUM + 1) - 1;                   PAR00560
800570   1  2        IF MATCHING (MACH.MATCH (N)) THEN DO;                        PAR00570
800580   1  3           STATE_NUM = ACTING (MACH.ACTION (N),MACH.NEXT_STATE (N)); PAR00580
800590   1  3           N = 10000;                                                PAR00590
800600   1  3           END;                                                      PAR00600
800610   1  2        END;                                                         PAR00610
800620   1  1     IF N < 10000 THEN DO;                                           PAR00620
                     /*TRANSITIONS NOT CLOSED*/                                   PAR00630
800640   1  2        PUT FILE (ERROR) SKIP EDIT                                   PAR00640
                        ('CLOSURE ON STATE = ',STATE_NUM,                         PAR00650
                         'INPUT =$',INPUT_PTR -> TOKEN.ITEM,'$$',                 PAR00660
                         'CLASS =$',INPUT_PTR -> TOKEN.CLASS,'$$',                PAR00670
                         'STK#1 =$', STACK (1,TOS (1)),'$$',                      PAR00680
                         'STK#2 =$', STACK (2,TOS (2)),'$$')                      PAR00690
                        (A,F(5),4(SKIP,A,A,A));                                   PAR00700
800710   1  2        IF DEBUG THEN                                                PAR00710
                        PUT SKIP LIST ('PREMATURE TERMINATION');                  PAR00720
800730   1  2        STATE_NUM = 0;                                               PAR00730
800740   1  2        END;                                                         PAR00740
800750   1  1     END;                                                            PAR00750
800760   1  0  CLOSE FILE (ERROR);                                                PAR00760
                                                                                  PAR00770
                                                                                  PAR00780
```

```
 NUMBER LEV NT

 800790   1  0  MATCHING: PROC (MATCH_LIST) RETURNS (BIT (1));                PAR00790
                                                                               PAR00800
 800810   2  0  DCL MATCH_LIST CHAR (30) VAR,                                  PAR00810
                    (REST_LIST,PROCESS_LIST) CHAR (30) VAR,                    PAR00820
                     I FIXED;                                                  PAR00830
                                                                               PAR00840
 800850   2  0  IF MATCH_LIST = '' THEN                                        PAR00850
                   RETURN ('1'B);                                              PAR00860
 800870   2  0  REST_LIST = MATCH_LIST;                                        PAR00870
 800880   2  0  DO WHILE (REST_LIST ^= '');                                    PAR00880
 800890   2  1     PROCESS_LIST = GETS (REST_LIST,'¦');                        PAR00890
 800900   2  1     IF DEBUG THEN                                               PAR00900
                      PUT SKIP EDIT                                            PAR00910
                         ('MATCHING =$',PROCESS_LIST,'$$')                     PAR00920
                         (3 (A));                                              PAR00930
 800940   2  1     IF FOUND (INPUT_PTR -> TOKEN.ITEM,                          PAR00940
                             INPUT_PTR -> TOKEN.CLASS) THEN                    PAR00950
                      IF FOUND (STACK (1,TOS (1)),'') THEN                     PAR00960
                         IF FOUND (STACK (2,TOS (2)),'') THEN                  PAR00970
                            RETURN ('1'B);                                     PAR00980
 800990   2  1     END;                                                        PAR00990
 801000   2  0  RETURN ('0'B);                                                 PAR01000
                                                                               PAR01010
```

```
NUMBER LEV NT

801020   2  0  FOUND: PROC (FIND_ITEM,FIND_CLASS) RETURNS (BIT (1));                 PAR01020
                                                                                     PAR01030
801040   3  0  DCL (FIND_ITEM,MATCH_ITEM,MATCH_FIND) CHAR (30) VAR,                  PAR01040
                    FIND_CLASS CHAR (10) VAR;                                        PAR01050
                                                                                     PAR01060
801070   3  0  MATCH_ITEM = GETS (PROCESS_LIST,',');                                 PAR01070
801080   3  0  IF DEBUG THEN                                                         PAR01080
                  PUT SKIP EDIT                                                      PAR01090
                     ('FIND_MATCH =$',MATCH_ITEM,'$$',                               PAR01100
                      'FIND_ITEM =$',FIND_ITEM,'$$',                                 PAR01110
                      'FIND_CLASS =$',FIND_CLASS,'$$')                               PAR01120
                     (A,A,A,2(SKIP,A,A,A));                                          PAR01130
801140   3  0  SELECT (MATCH_ITEM);                                                  PAR01140
801150   3  1     WHEN ('',FIND_CLASS) RETURN ('1'B);                                PAR01150
801160   3  1     WHEN ('@CONC') MATCH_ITEM = '¦';                                   PAR01160
801170   3  1     WHEN ('@CONC>') MATCH_ITEM = '¦,+,-,*,/,!';                        PAR01170
801180   3  1     WHEN ('@SUMOP') MATCH_ITEM = '+,-';                                PAR01180
801190   3  1     WHEN ('@SUMOP>') MATCH_ITEM = '+,-,*,/,!';                         PAR01190
801200   3  1     WHEN ('@MULTOP') MATCH_ITEM = '*,/';                               PAR01200
801210   3  1     WHEN ('@MULTOP>') MATCH_ITEM = '*,/,**';                           PAR01210
801220   3  1     WHEN ('@VIRT') MATCH_ITEM = 'VO,V1,V2,V3,V4,V5,V6,V7,V8,V9';       PAR01220
801230   3  1     WHEN ('@SETOP') MATCH_ITEM = 'MU,MI,SU,SI,CS';                     PAR01230
801240   3  1     WHEN ('@REL') MATCH_ITEM = '>,<,=';                                PAR01240
801250   3  1     WHEN ('@CMP') MATCH_ITEM = 'AND,OR,XOR';                           PAR01250
801260   3  1     OTHERWISE;                                                         PAR01260
801270   3  1     END;                                                               PAR01270
801280   3  0  MATCH_FIND = FIND_ITEM;                                               PAR01280
801290   3  0  MATCH_FIND = GETS (MATCH_FIND,':');                                   PAR01290
801300   3  0  DO WHILE (MATCH_ITE. ^= '');                                          PAR01300
801310   3  1     IF DEBUG THEN                                                      PAR01310
                     PUT SKIP EDIT ('SEP_ITEM =$',MATCH_ITEM,'$$')                   PAR01320
                                   (3 (A));                                          PAR01330
801340   3  1     IF GETS (MATCH_ITEM,',') = MATCH_FIND THEN                         PAR01340
                     RETURN ('1'B);                                                  PAR01350
801360   3  1     END;                                                               PAR01360
801370   3  0  RETURN ('0'B);                                                        PAR01370
                                                                                     PAR01380
801390   3  0  END FOUND;                                                            PAR01390
                                                                                     PAR01400
801410   2  0  END MATCHING;                                                         PAR01410
                                                                                     PAR01420
```

```
NUMBER LEV NT

801430   1  0  ACTING: PROC (ACTION_LIST,NXT_STATE) RETURNS (FIXED);               PAR01430
                                                                                  PAR01440
801450   2  0  DCL (ACTION_LIST,REST_LIST) CHAR (40) VAR,                         PAR01450
                  (PROCESS_LIST,ACTION_NAME,GARBAGE,RTN_STATE) CHAR (30) VAR,    PAR01460
                  NXT_STATE FIXED;                                                PAR01470
                                                                                  PAR01480
801490   2  0  REST_LIST = ACTION_LIST;                                           PAR01490
801500   2  0  DO WHILE (REST_LIST ^= '');                                        PAR01500
801510   2  1     PROCESS_LIST = GETS (REST_LIST,'|');                            PAR01510
801520   2  1     IF DEBUG THEN                                                   PAR01520
                     PUT SKIP EDIT                                                PAR01530
                        ('ACTING =$',PROCESS_LIST,'$$')                           PAR01540
                        (3 (A));                                                  PAR01550
801560   2  1     ACTION_NAME = GETS (PROCESS_LIST,',');                          PAR01560
801570   2  1     SELECT (ACTION_NAME);                                           PAR01570
801580   2  2        WHEN ('PUSH','P')                                            PAR01580
                        CALL PUSH (FIXED (GETS (PROCESS_LIST,',')),               PAR01590
                                   CHANGE (PROCESS_LIST));                        PAR01600
801610   2  2        WHEN ('POP')                                                 PAR01610
                        GARBAGE = POP (FIXED (PROCESS_LIST));                     PAR01620
801630   2  2        WHEN ('DEL') DO;                                             PAR01630
801640   2  3           FREE_PTR = INPUT_PTR;                                     PAR01640
801650   2  3           INPUT_PTR = INPUT_PTR -> TOKEN.NEXT;                      PAR01650
801660   2  3           FREE FREE_PTR -> TOKEN;                                   PAR01660
801670   2  3           END;                                                      PAR01670
801680   2  2        WHEN ('GENENT')                                              PAR01680
                        CALL GENENT;                                              PAR01690
801700   2  2        WHEN ('VIRTX')                                               PAR01700
                        CALL VIRTX;                                               PAR01710
801720   2  2        WHEN ('VIRTA')                                               PAR01720
                        CALL VIRTA;                                               PAR01730
801740   2  2        WHEN ('ATTWHR')                                              PAR01740
                        ENTITY (CUR_ES).WHERE = FIXED (SUBSTR (POP (1),2));       PAR01750
801760   2  2        WHEN ('GENNODE','GD')                                        PAR01760
                        CALL GENNODE;                                             PAR01770
801780   2  2        WHEN ('INDX')                                                PAR01780
                        CALL EXCH (':IND');                                       PAR01790
801800   2  2        WHEN ('MULX')                                                PAR01800
                        CALL EXCH (':MUL');                                       PAR01810
801820   2  2        WHEN ('ADDON')                                               PAR01820
                        CALL ADDON;                                               PAR01830
801840   2  2        OTHERWISE                                                    PAR01840
                        PUT FILE (ERROR) SKIP LIST                                PAR01850
                           ('BAD ACTION: ',ACTION_NAME);                          PAR01860
801870   2  2        END;                                                         PAR01870
801880   2  1     END;                                                            PAR01880
801890   2  0     IF NXT_STATE < 0 THEN DO;                                       PAR01890
801900   2  1        RTN_STATE = POP (2);                                         PAR01900
801910   2  1        GARBAGE = GETS (RTN_STATE,':');                              PAR01910
```

```
 NUMBER LEV NT

 801920   2  1         RETURN (FIXED (RTN_STATE));                                          PAR01920
 801930   2  1         END;                                                                 PAR01930
 801940   2  0      ELSE                                                                    PAR01940
                       RETURN (NXT_STATE);                                                  PAR01950
                                                                                            PAR01960
 801970   2  0   END ACTING;                                                                PAR01970
                                                                                            PAR01980
```

```
NUMBER LEV NT

801990   1  0  GETS: PROC (LIST,TERM_ITEM) RETURNS (CHAR (30) VAR);       PAR01990
                                                                           PAR02000
802010   2  0  DCL LIST CHAR (*) VAR,                                      PAR02010
                   TERM_ITEM CHAR (1),                                     PAR02020
                   RTN_LIST CHAR (30) VAR,                                 PAR02030
                   I FIXED;                                                PAR02040
                                                                           PAR02050
802060   2  0  I = INDEX (LIST,TERM_ITEM);                                 PAR02060
802070   2  0  IF I = 0 THEN DO;                                           PAR02070
802080   2  1     RTN_LIST = LIST;                                         PAR02080
802090   2  1     LIST = '';                                               PAR02090
802100   2  1     END;                                                     PAR02100
802110   2  0  ELSE DO;                                                    PAR02110
802120   2  1     RTN_LIST = SUBSTR (LIST,1,I - 1);                        PAR02120
802130   2  1     LIST = SUBSTR (LIST,I + 1);                              PAR02130
802140   2  1     END;                                                     PAR02140
802150   2  0  RETURN (RTN_LIST);                                          PAR02150
                                                                           PAR02160
802170   2  0  END GETS;                                                   PAR02170
                                                                           PAR02180
```

```
NUMBER LEV NT

802190   1  0  PUSH: PROC (STACK_NUM,STACK_ITEM);

802210   2  0  DCL STACK_NUM FIXED,
                   STACK_ITEM CHAR (30) VAR;

802240   2  0  TOS (STACK_NUM) = TOS (STACK_NUM) + 1;
802250   2  0  STACK (STACK_NUM,TOS (STACK_NUM)) = STACK_ITEM;

802270   2  0  END PUSH;


802290   1  0  POP: PROC (STACK_NUM) RETURNS (CHAR (30) VAR);

802310   2  0  DCL STACK_NUM FIXED;

802330   2  0  TOS (STACK_NUM) = TOS (STACK_NUM) - 1;
802340   2  0  RETURN (STACK (STACK_NUM,TOS (STACK_NUM) + 1));

802360   2  0  END POP;


802380   1  0  CHANGE: PROC (CH_ITEM) RETURNS (CHAR (30) VAR);

802400   2  0  DCL (CH_ITEM,INP_SOURCE) CHAR (30) VAR;

802420   2  0  IF INDEX (CH_ITEM,'@') = 0 THEN
                  RETURN (CH_ITEM);
802440   2  0  INP_SOURCE = GETS (CH_ITEM,'@');
802450   2  0  SELECT (INP_SOURCE);
802460   2  1     WHEN ('I')
                     CH_ITEM = INPUT_PTR -> TOKEN.ITEM || CH_ITEM;
802480   2  1     WHEN ('C')
                     CH_ITEM = INPUT_PTR -> TOKEN.ITEM
                            || CH_ITEM || INPUT_PTR -> TOKEN.CLASS;
802510   2  1     WHEN ('1','2','3')
                     CH_ITEM = STACK (FIXED (INP_SOURCE),TOS (FIXED (INP_SOURCE)))
                             || CH_ITEM;
802540   2  1     OTHERWISE;
802550   2  1     END;
802560   2  0  RETURN (CH_ITEM);

802580   2  0  END CHANGE;
```

AD-A116 503 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA F/G 9/2
VIRTUAL INFORMATION FACILITY OF THE INFOPLEX SOFTWARE TEST VEHI--ETC(U)
MAY 82 P LU N00039-81-C-0663
UNCLASSIFIED M010-8205-11 NL

2 OF 2
AD A
116503

END
DATE
FILMED
07-82
DTIC

```
NUMBER LEV NT

802600   1  0  GENENT: PROC;                                                            PAR02600
                                                                                        PAR02610
802620   2  0  DCL (ENT_ITEM,ENT_NAME) CHAR (30) VAR,                                   PAR02620
                    (NUM_KEYS,I) FIXED;                                                 PAR02630
                                                                                        PAR02640
802650   2  0  CUR_ES = CUR_ES + 1;                                                     PAR02650
802660   2  0  ENTITY (CUR_ES).VES_FN = 'S';                                            PAR02660
802670   2  0  ENT_ITEM = POP (2);                              /*EITHER "(" OR ")"*/   PAR02670
802680   2  0  IF ENT_ITEM = ')' THEN DO;                                               PAR02680
802690   2  1     ENTITY (CUR_ES).N_PARENT (2) = FIXED (SUBSTR (POP (1),2));            PAR02690
802700   2  1     NUM_KEYS = 0;                                                         PAR02700
802710   2  1     DO WHILE (ENT_ITEM ^= '(');                                           PAR02710
802720   2  2        CALL PUSH (3,POP (1));                                             PAR02720
802730   2  2        NUM_KEYS = NUM_KEYS + 1;                                           PAR02730
802740   2  2        ENT_ITEM = POP (2);                                                PAR02740
802750   2  2        END;                                                               PAR02750
802760   2  1     DO I = 1 TO NUM_KEYS;                                                 PAR02760
802770   2  2        CALL GENATTR (POP (3));                                            PAR02770
802780   2  2        END;                                                               PAR02780
802790   2  1     ENTITY (CUR_ES).VES_FN = POP (2);                                     PAR02790
802800   2  1     ENT_ITEM = POP (2);                           /*MUST BE "("*/         PAR02800
802810   2  1     END;                                                                  PAR02810
802820   2  0  ENT_NAME = POP (1);                                                      PAR02820
802830   2  0  IF SUBSTR (ENT_NAME,1,1) = ':' THEN                                      PAR02830
                  ENTITY (CUR_ES).N_PARENT (1) = FIXED (SUBSTR (ENT_NAME,2));           PAR02840
802850   2  0  ELSE DO;                                                                 PAR02850
802860   2  1     ENTITY (CUR_ES).NAME = ENT_NAME;                                      PAR02860
802870   2  1     ENTITY (CUR_ES).VES_FN = 'R';                                         PAR02870
802880   2  1     END;                                                                  PAR02880
802890   2  0  CALL PUSH (1,':' || CHAR (CUR_ES));                                      PAR02890
                                                                                        PAR02900
802910   2  0  END GENENT;                                                              PAR02910
                                                                                        PAR02920
```

```
NUMBER LEV NT

802930    1  0  VIRTX: PROC;

802950    2  0  DCL GARBAGE CHAR (30) VAR;

802970    2  0  STACK (1,TOS (1)) =
                   ':' || VES (FIXED (SUBSTR (STACK (1,TOS (1)),2)));
                                                          /*MUST BE "{"*/
802990    2  0  GARBAGE = POP (2);

803010    2  0  END VIRTX;

803030    1  0  VIRTA: PROC;

803050    2  0  VES_LOC = VES_LOC + 1;
803060    2  0  VES (VES_LOC) = FIXED (SUBSTR (POP (1),2));

803080    2  0  END VIRTA;
```

```
NUMBER LEV NT

803100   1  0  GENNODE: PROC;

803120   2  0  DCL (OP_ITEM,ARG_ITEM,NODE_LOCATION) CHAR (30) VAR,
                   I FIXED;

803150   2  0  OP_ITEM = POP (2);
803160   2  0  NODE_LOCATION = ':' || CHAR (XTREE_LOC);
803170   2  0  CALL PUTX (GETS (OP_ITEM,':'),FIXED (OP_ITEM));
803180   2  0  IF FIXED (OP_ITEM) > 0 THEN                      /*0 IF NO ARG OP*/
                  DO I = 1 TO FIXED (OP_ITEM);
803200   2  1        ARG_ITEM = POP (1);
803210   2  1        IF SUBSTR (ARG_ITEM,1,1) = ':' THEN
                        XTREE (FIXED (POP (3))).LINK = FIXED (SUBSTR (ARG_ITEM,2));
803230   2  1        ELSE DO;
803240   2  2           XTREE (FIXED (POP (3))).LINK = XTREE_LOC;
803250   2  2           CALL GENATTR (ARG_ITEM);
803260   2  2           CALL PUTX (ARG_ITEM,0);
803270   2  2           END;
803280   2  1        END;
803290   2  0  CALL PUSH (1,NODE_LOCATION);


803310   2  0  PUTX: PROC (PUTX_ITEM,CHILD_NUM);

803330   3  0  DCL PUTX_ITEM CHAR (30) VAR,
                     (CHILD_NUM,I) FIXED;

803360   3  0  XTREE (XTREE_LOC).LABEL = PUTX_ITEM;
803370   3  0  XTREE (XTREE_LOC).CHILD = CHILD_NUM;
803380   3  0  IF CHILD_NUM > 0 THEN
                  DO I = 0 TO CHILD_NUM - 1;
803400   3  1        CALL PUSH (3,CHAR (XTREE_LOC + I));  /*WATCH FOR CONV ERRORS*/
803410   3  1        END;
803420   3  0  XTREE_LOC = XTREE_LOC + MAX (CHILD_NUM,1);

803440   3  0  END PUTX;

803460   2  0  END GENNODE;
```

```
NUMBER LEV NT

803480   1  0  GENATTR: PROC (ATTR_ITEM);                                              PAR03480
                                                                                       PAR03490
               /*MODIFIES ATTR_ITEM*/                                                  PAR03500
803510   2  0  DCL (ATTR_ITEM,GEN_ITEM,VES_CART_KEY,NM_ITEM) CHAR (30) VAR,            PAR03510
                    NAME_ITEM CHAR (80) VAR,                                           PAR03520
                    (PARENT_NUM,I) FIXED,                                              PAR03530
                    (VIRTUAL,CARTESIAN) BIT (1);                                       PAR03540
                                                                                       PAR03550
803560   2  0  GEN_ITEM = ATTR_ITEM;                                                   PAR03560
803570   2  0  NAME_ITEM = GETS (GEN_ITEM,':');                                        PAR03570
803580   2  0  IF GEN_ITEM = 'IND' THEN                                                PAR03580
                  NAME_ITEM = XCHNGE (FIXED (NAME_ITEM));                              PAR03590
803600   2  0  ELSE IF GEN_ITEM ^= 'R' THEN                                            PAR03600
                  RETURN;                                                              PAR03610
803620   2  0  PARENT_NUM = 0;                                                         PAR03620
803630   2  0  VIRTUAL = '0'B;                                                         PAR03630
803640   2  0  CARTESIAN = '0'B;                                                       PAR03640
803650   2  0  DO WHILE (NAME_ITEM ^= '');                                             PAR03650
803660   2  1     VES_CART_KEY = GETS (NAME_ITEM,',');                                 PAR03660
803670   2  1     NM_ITEM = GETS (VES_CART_KEY,';');                                   PAR03670
803680   2  1     VIRTUAL = ((GETS (VES_CART_KEY,';') = 'V') | VIRTUAL);               PAR03680
803690   2  1     CARTESIAN = ((VES_CART_KEY = 'C') | CARTESIAN);                      PAR03690
803700   2  1     DO I = PARENT_NUM + 1 TO 15                                          PAR03700
                     WHILE (ENTITY (CUR_ES).ATTR (I).USES ^= '');                      PAR03710
803720   2  2        IF NM_ITEM = ENTITY (CUR_ES).ATTR (I).USES                        PAR03720
                        & CARTESIAN = ENTITY (CUR_ES).ATTR (I).CART_KEY                PAR03730
                        & PARENT_NUM = ENTITY (CUR_ES).ATTR (I).A_PARENT THEN DO;      PAR03740
803750   2  3           PARENT_NUM = I;                                                PAR03750
803760   2  3           I = 20;                                                        PAR03760
803770   2  3           END;                                                           PAR03770
803780   2  2        END;                                                              PAR03780
803790   2  1     IF I = 16 THEN                                                       PAR03790
                     /*TOO MAY ATTRIBUTES FOR THIS ES*/;                               PAR03800
803810   2  1     ELSE IF I < 16 THEN DO;                                              PAR03810
803820   2  2        ENTITY (CUR_ES).ATTR (I).USES = NM_ITEM;                          PAR03820
803830   2  2        ENTITY (CUR_ES).ATTR (I).VES_KEY = VIRTUAL;                       PAR03830
803840   2  2        ENTITY (CUR_ES).ATTR (I).CART_KEY = CARTESIAN;                    PAR03840
803850   2  2        ENTITY (CUR_ES).ATTR (I).A_PARENT = PARENT_NUM;                   PAR03850
803860   2  2        PARENT_NUM = I;                                                   PAR03860
803870   2  2        END;                                                              PAR03870
803880   2  1     END;                                                                 PAR03880
803890   2  0     ATTR_ITEM = CHAR (PARENT_NUM) || ':NTH';                             PAR03890
803900   2  0  END GENATTR;                                                            PAR03900
                                                                                       PAR03910
```

```
NUMBER LEV NT

803920   1  0  EXCH: PROC (TYP);

803940   2  0  DCL TYP CHAR (4);

803960   2  0  XCH_LOC = XCH_LOC + 1;
803970   2  0  XCHNGE (XCH_LOC) = GETS (STACK (1,TOS (1)),':');
803980   2  0  STACK (1,TOS (1)) = CHAR (XCH_LOC) || TYP;

804000   2  0  END EXCH;

804020   1  0  ADDON: PROC;

804040   2  0  XCHNGE (XCH_LOC) = XCHNGE (XCH_LOC) || ',' || STACK (1,TOS (1));

804060   2  0  END ADDON;
```

```
NUMBER LEV NT

804080   1  0  END PARSE;                                    PAR04080
```

```
NUMBER LEV NT

     10        0  SMPLFY: PROC (XTREE,ENTITY,XCHNGE,RETPTR);                        SMP00010
                                                                                    SMP00020
                  %INCLUDE XTREE;*************************************************SMP00030
 100010   1   0  DCL 1 XTREE (1000),                                               XTR00010
                        2 LABEL CHAR (30) VAR,                                      XTR00020
                        2 CHILD FIXED,                                              XTR00030
                        2 LINK FIXED;                                               XTR00040
                  ***************                                                   SMP00030
                  %INCLUDE XCHNGE;************************************************SMP00040
 300010   1   0  DCL XCHNGE (20) CHAR (80) VAR;                                    IND00010
                  ****************                                                  SMP00040
                  %INCLUDE ENTITY;************************************************SMP00050
 400010   1   0  DCL 1 ENTITY (0:12),                                              ENT00010
                           2 NAME CHAR (30) VAR,                                    ENT00020
                           2 DEPTH FIXED,                                           ENT00030
                           2 VES_FN CHAR (2) VAR,                                   ENT00040
                           2 WHERE FIXED,                                           ENT00050
                           2 N_PARENT (2) FIXED,                                    ENT00060
                           2 VES_PAR PTR,                                           ENT00070
                           2 ATTR (15),                                             ENT00080
                              3 VES_KEY BIT (1),                                    ENT00090
                              3 CART_KEY BIT (1),                                   ENT00100
                              3 SING_OCC BIT (1),                                   ENT00110
                              3 A_PARENT FIXED,                                     ENT00120
                              3 USES CHAR (30) VAR,                                 ENT00130
                              3 LIST PTR;                                           ENT00140
 400150   1   0  DCL 1 N_MAP (2) BASED,                                            ENT00150
                           2 NUM (15) FIXED;                                        ENT00160
                  ****************                                                  SMP00050
                  %INCLUDE ATTRIB;************************************************SMP00060
 600010   1   0  DCL 1 ATTRIB BASED,                                               ATT00010
                        2 LEVEL FIXED,                                              ATT00020
                        2 ITEM CHAR (50) VAR,                                       ATT00030
                        2 NEXT PTR;                                                 ATT00040
                  ****************                                                  SMP00060
                  %INCLUDE ARETE;*************************************************SMP00070
 700020   1   0  DCL 1 RETE_ARG BASED (RETEP),  /* WILL ALSO BE RETE_RTN */        F2A00020
                          2 CTL_INFO,                                               F2A00030
                            3 LEN FIXED BIN (15),                                   F2A00040
                            3 CBTP FIXED BIN (15) INIT (21),                        F2A00050
                            3 PTR PTR,                                              F2A00060
                          2 NUM_ATTR FIXED,                                         F2A00070
                          2 NUM_COND FIXED,    /* NUMBER OF CONDITIONS */           F2A00080
                          2 ENT,                                                    F2A00090
                            3 NAME CHAR (30) VAR,  /* ENTITY SET NAME */            F2A00100
                            3 DEPTH FIXED,     /* FILLED IN WHEN RETURNED */        F2A00110
                            3 ATTR (NATTR REFER (RETE_ARG.NUM_ATTR)),               F2A00120
                              4 SING_OCC BIT (1), /* IF SINGLE OCCUR THEN SET */    F2A00130
                              4 A_PARENT FIXED, /* PARENT NUMBER */                 F2A00140
```

```
 NUMBER LEV NT

                          4 USES CHAR(30) VAR, /* ATTRIBUTE NAME */              F2A00150
                          4 LIST PTR, /* POINT TO LIST OF OCC OF THIS ATTR IF ANY */  F2A00160
                        2 COND (NCOND REFER (RETE_ARG.NUM_COND)),                F2A00170
                         3 ATTRREF FIXED, /* POINT TO ATTR IN ATTR ARRAY ABOVE */    F2A00180
                         3 NEG BIT(1), /* NEGATION OF RELATOR */                  F2A00190
                         3 REL CHAR(1), /* '=', '<', '>' */                       F2A00200
                         3 CDATA (10) CHAR (50) VAR,  /* UP TO 10 'MULTI' ITMES */ F2A00210
                        2 RTN_CODE FIXED BIN,                                    F2A00220
                        2 NEXT_PTR PTR;                                          F2A00230
 700240   1  0  DCL RETEP PTR;                                                   F2A00240
                                                                                 F2A00250
 700260   1  0  DCL 1 RETE_RTN1 BASED,          /* USED WHEN RETURNED */         F2A00260
                        2 CTL,                                                   F2A00270
                         3 LEN FIXED BIN(31),                                    F2A00280
                         3 CBTP FIXED BIN (31) INIT (46),                        F2A00290
                         3 PTR PTR,                                              F2A00300
                        2 LEVEL FIXED BIN, /* OCCUR NUMBER */                    F2A00310
                        2 ITEM CHAR (50),                                        F2A00320
                        2 NEXT_PTR PTR; /* POINT TO NEXT ON THE SAME ATTRIBUTE LIST */  F2A00330
                ***************                                                  SMP00070
                                                                                 SMP00080
 800090   1  0  DCL (RETPTR,TAILR,P) PTR,                                        SMP00090
                      (CUR_ES,NCOND,NATTR,I) FIXED;                              SMP00100
 800110   1  0  DCL 1 ENTCOND (12,12),                                           SMP00110
                      2 ATTRREF FIXED,                                           SMP00120
                      2 NEG BIT (1),                                             SMP00130
                      2 REL CHAR (1) VAR,                                        SMP00140
                      2 CDATA (10) CHAR (30) VAR;                                SMP00150
                                                                                 SMP00160
 800170   1  0  ENTCOND (*,*).ATTRREF = 0;                                       SMP00170
 800180   1  0  ENTCOND (*,*).NEG = '0'B;                                        SMP00180
 800190   1  0  ENTCOND (*,*).REL = '';                                          SMP00190
 800200   1  0  ENTCOND (*,*).CDATA (*) = '';                                    SMP00200
                                                                                 SMP00210
```

100

```
NUMBER LEV NT

800220   1  0  DO CUR_ES = 12 TO 1 BY -1;                                              SMP00220
800230   1  1     SELECT (ENTITY (CUR_ES).VES_FN);                                      SMP00230
800240   1  2        WHEN ('S') DO;                                                     SMP00240
800250   1  3           ALLOCATE N_MAP SET (ENTITY (CUR_ES).VES_PAR);                   SMP00250
800260   1  3           CALL PRPGATE (ENTITY (CUR_ES).N_PARENT (1),                     SMP00260
                                      'O'B,                                             SMP00270
                                      ENTITY (CUR_ES).VES_PAR -> N_MAP (1).NUM (*));   SMP00280
800290   1  3           END;                                                            SMP00290
800300   1  2        WHEN ('MU','MI','SU','SI') DO;                                     SMP00300
800310   1  3           ALLOCATE N_MAP SET (ENTITY (CUR_ES).VES_PAR);                   SMP00310
800320   1  3           CALL PRPGATE (ENTITY (CUR_ES).N_PARENT (1),                     SMP00320
                                      'O'B,                                             SMP00330
                                      ENTITY (CUR_ES).VES_PAR -> N_MAP (1).NUM (*));   SMP00340
800350   1  3           CALL PRPGATE (ENTITY (CUR_ES).N_PARENT (2),                     SMP00350
                                      'O'B,                                             SMP00360
                                      ENTITY (CUR_ES).VES_PAR -> N_MAP (2).NUM (*));   SMP00370
800380   1  3           END;                                                            SMP00380
800390   1  2        WHEN ('CS') DO;                                                    SMP00390
800400   1  3           ALLOCATE N_MAP SET (ENTITY (CUR_ES).VES_PAR);                   SMP00400
800410   1  3           CALL PRPGATE (ENTITY (CUR_ES).N_PARENT (1),                     SMP00410
                                      '1'B,                                             SMP00420
                                      ENTITY (CUR_ES).VES_PAR -> N_MAP (1).NUM (*));   SMP00430
800440   1  3           CALL PRPGATE (ENTITY (CUR_ES).N_PARENT (2),                     SMP00440
                                      'O'B,                                             SMP00450
                                      ENTITY (CUR_ES).VES_PAR -> N_MAP (2).NUM (*));   SMP00460
800470   1  3           END;                                                            SMP00470
800480   1  2        OTHERWISE;                                                         SMP00480
800490   1  2        END;                                                               SMP00490
800500   1  1     END;                                                                  SMP00500
                                                                                        SMP00510
```

```
NUMBER LEV NT

800520   1  0  RETPTR = NULL ();                                                      SMP00520
800530   1  0  DO CUR_ES = 1 TO 12 WHILE (ENTITY (CUR_ES).VES_FN ^= '');                SMP00530
800540   1  1     NCOND = 0;                                                          SMP00540
800550   1  1     IF ENTITY (CUR_ES).VES_FN = 'R' THEN DO;                            SMP00550
800560   1  2        IF ENTITY (CUR_ES).WHERE > 0                                     SMP00560
                        & SEARCH (ENTITY (CUR_ES).WHERE) <= -3 THEN                   SMP00570
                        ENTITY (CUR_ES).WHERE = 0;                                    SMP00580
800590   1  2        DO I = 1 TO 15 WHILE (ENTITY (CUR_ES).ATTR (I).USES ^= '');       SMP00590
800600   1  3           END;                                                          SMP00600
800610   1  2        NATTR = I - 1;                                                   SMP00610
800620   1  2        ALLOCATE RETE_ARG SET (P);                                       SMP00620
800630   1  2        P -> RETE_ARG.ENT.NAME = ENTITY (CUR_ES).NAME;                   SMP00630
800640   1  2        P -> RETE_ARG.ENT.DEPTH = ENTITY (CUR_ES).DEPTH;                 SMP00640
800650   1  2        DO I = 1 TO NATTR;                                               SMP00650
800660   1  3           P -> RETE_ARG.ENT.ATTR (I).SING_OCC =                         SMP00660
                           ENTITY (CUR_ES).ATTR (I).SING_OCC;                         SMP00670
800680   1  3           P -> RETE_ARG.ENT.ATTR (I).A_PARENT =                         SMP00680
                           ENTITY (CUR_ES).ATTR (I).A_PARENT;                         SMP00690
800700   1  3           P -> RETE_ARG.ENT.ATTR (I).USES =                             SMP00700
                           ENTITY (CUR_ES).ATTR (I).USES;                             SMP00710
800720   1  3           END;                                                          SMP00720
800730   1  2        DO I = 1 TO NCOND;                                               SMP00730
800740   1  3           P -> RETE_ARG.COND (I).ATTRREF = ENTCOND (CUR_ES,I).ATTRREF;  SMP00740
800750   1  3           P -> RETE_ARG.COND (I).NEG = ENTCOND (CUR_ES,I).NEG;          SMP00750
800760   1  3           P -> RETE_ARG.COND (I).REL = ENTCOND (CUR_ES,I).REL;          SMP00760
800770   1  3           P -> RETE_ARG.COND (I).CDATA (*) =                            SMP00770
                           ENTCOND (CUR_ES,I).CDATA (*);                              SMP00780
800790   1  3           END;                                                          SMP00790
800800   1  2        IF RETPTR = NULL () THEN                                         SMP00800
                        RETPTR = P;                                                   SMP00810
800820   1  2        ELSE DO;                                                         SMP00820
800830   1  3           TAILR -> RETE_ARG.NEXT_PTR = P;                               SMP00830
800840   1  3           TAILR -> RETE_ARG.CTL_INFO.PTR = P;                           SMP00840
800850   1  3           END;                                                          SMP00850
800860   1  2        TAILR = P;                                                       SMP00860
800870   1  2        END;                                                             SMP00870
800880   1  1     TAILR -> RETE_ARG.NEXT_PTR = NULL ();                               SMP00880
800890   1  1     TAILR -> RETE_ARG.CTL_INFO.PTR = NULL ();                           SMP00890
800900   1  1     END;                                                                SMP00900
                                                                                      SMP00910
```

```
NUMBER LEV NT

800920   1  0  PRPGATE: PROC (PAR_ENT,CART_TYP,MAP_NUM);                       SMP00920
                                                                               SMP00930
800940   2  0  DCL PAR_ENT FIXED;                                              SMP00940
800950   2  0  DCL CART_TYP BIT (1);                                           SMP00950
800960   2  0  DCL MAP_NUM (15) FIXED;                                         SMP00960
800970   2  0  DCL (I,J,PARENT_NUM) FIXED;                                     SMP00970
800980   2  0  DCL NAME_ITEM CHAR (80) VAR,                                    SMP00980
                   NM_ITEM CHAR (30) VAR;                                      SMP00990
                                                                               SMP01000
801010   2  0  MAP_NUM (*) = 0;                                                SMP01010
801020   2  0  DO I = 1 TO 15 WHILE (ENTITY (CUR_ES).ATTR (I).USES ^= '');     SMP01020
801030   2  1     IF ENTITY (CUR_ES).ATTR (I).CART_KEY = CART_TYP THEN DO;     SMP01030
801040   2  2        J = I;                                                    SMP01040
801050   2  2        NAME_ITEM = ENTITY (CUR_ES).ATTR (J).USES;                SMP01050
801060   2  2        DO WHILE (ENTITY (CUR_ES).ATTR (J).A_PARENT ^= 0);        SMP01060
801070   2  3           J = ENTITY (CUR_ES).ATTR (J).A_PARENT;                 SMP01070
801080   2  3           NAME_ITEM = ENTITY (CUR_ES).ATTR (J).USES || ','       SMP01080
                                    || NAME_ITEM;                              SMP01090
801100   2  3           END;                                                   SMP01100
801110   2  2        PARENT_NUM = 0;                                           SMP01110
801120   2  2        DO WHILE (NAME_ITEM ^= '');                               SMP01120
801130   2  3           NM_ITEM = GETS (NAME_ITEM,',');                        SMP01130
801140   2  3           DO J = PARENT_NUM + 1 TO 15                            SMP01140
                           WHILE (ENTITY (PAR_ENT).ATTR (J).USES ^= '');       SMP01150
801160   2  4              IF NM_ITEM = ENTITY (PAR_ENT).ATTR (J).USES         SMP01160
                              & ENTITY (PAR_ENT).ATTR (J).CART_KEY = '0'B      SMP01170
                              & PARENT_NUM = ENTITY (PAR_ENT).ATTR (J).A_PARENT THEN SMP01180
                              DO;                                              SMP01190
801200   2  5                 PARENT_NUM = J;                                  SMP01200
801210   2  5                 J = 20;                                          SMP01210
801220   2  5                 END;                                             SMP01220
801230   2  4              END;                                                SMP01230
801240   2  3           IF J = 16 THEN                                         SMP01240
                           /*TOO MAY ATTRIBUTES FOR THIS ES*/;                 SMP01250
801260   2  3           ELSE IF J < 16 THEN DO;                                SMP01260
801270   2  4              ENTITY (PAR_ENT).ATTR (J).USES = NM_ITEM;           SMP01270
801280   2  4              ENTITY (PAR_ENT).ATTR (J).VES_KEY = '0'B;           SMP01280
801290   2  4              ENTITY (PAR_ENT).ATTR (J).CART_KEY = '0'B;;         SMP01290
801300   2  4              ENTITY (PAR_ENT).ATTR (J).A_PARENT = PARENT_NUM;    SMP01300
801310   2  4              PARENT_NUM = J;                                     SMP01310
801320   2  4              END;                                                SMP01320
801330   2  3           END;                                                   SMP01330
801340   2  2        MAP_NUM (I) = PARENT_NUM;                                 SMP01340
801350   2  2        END;                                                      SMP01350
801360   2  1     END;                                                         SMP01360
                                                                               SMP01370
801380   2  0  END PRPGATE;                                                    SMP01380
                                                                               SMP01390
```

```
 NUMBER LEV NT

 801400   1  0  SEARCH: PROC (LINK_PT) RETURNS (FIXED) RECURSIVE;                 SMP01400
                                                                                  SMP01410
 801420   2  0  DCL (LINK_PT,TAG1,TAG2) FIXED,                                    SMP01420
                     (TERM_TYP,TERM_NM) CHAR (30) VAR;                            SMP01430
                                                                                  SMP01440
 801450   2  0  IF XTREE (LINK_PT).CHILD = 0 THEN DO;                             SMP01450
 801460   2  1     TERM_TYP = XTREE (LINK_PT).LABEL;                              SMP01460
 801470   2  1     TERM_NM = GETS (TERM_TYP,':');                                 SMP01470
 801480   2  1     IF TERM_TYP = 'NTH' THEN                                       SMP01480
                      RETURN (-2);                                                SMP01490
 801500   2  1     ELSE IF INDEX ('A,S,MUL',TERM_TYP) ^= 0 THEN                   SMP01500
                      RETURN (-1);                                                SMP01510
 801520   2  1     END;                                                           SMP01520
 801530   2  0  ELSE IF XTREE (LINK_PT).CHILD = 1                                 SMP01530
                   & XTREE (LINK_PT).LABEL = '^'                                  SMP01540
                   & SEARCH (XTREE (LINK_PT).LINK) = -3 THEN DO;                  SMP01550
 801560   2  1        ENTCOND (CUR_ES,NCOND).NEG =                                SMP01560
                         ^ENTCOND (CUR_ES,NCOND).NEG;                             SMP01570
 801580   2  1        RETURN (-3);                                                SMP01580
 801590   2  1     END;                                                           SMP01590
 801600   2  0  ELSE IF XTREE (LINK_PT).CHILD = 2                                 SMP01600
                   & INDEX ('>,<,=,AND',XTREE (LINK_PT).LABEL) ^= 0 THEN DO;      SMP01610
 801620   2  1     TAG1 = SEARCH (XTREE (LINK_PT).LINK);                          SMP01620
 801630   2  1     TAG2 = SEARCH (XTREE (LINK_PT + 1).LINK);                      SMP01630
 801640   2  1     IF TAG1 = -2 & TAG2 = -1 THEN DO;                              SMP01640
 801650   2  2        CALL SETREL (LINK_PT,                                       SMP01650
                                   XTREE (LINK_PT).LINK,                          SMP01660
                                   XTREE (LINK_PT + 1).LINK);                     SMP01670
 801680   2  2        RETURN (-3);                                                SMP01680
 801690   2  2        END;                                                        SMP01690
 801700   2  1     ELSE IF TAG1 = -1 & TAG2 = -2 THEN DO;                         SMP01700
 801710   2  2        CALL SETREL (LINK_PT,                                       SMP01710
                                   XTREE (LINK_PT + 1).LINK,                      SMP01720
                                   XTREE(LINK_PT).LINK);                          SMP01730
 801740   2  2        RETURN (-3);                                                SMP01740
 801750   2  2        END;                                                        SMP01750
 801760   2  1     ELSE IF TAG1 <= -3 THEN                                        SMP01760
                      IF TAG2 <= -3 THEN                                          SMP01770
                         RETURN (-4);                                             SMP01780
 801790   2  1        ELSE IF TAG2 = 0 THEN                                       SMP01790
                         RETURN (XTREE (LINK_PT + 1).LINK);                       SMP01800
 801810   2  1        ELSE                                                        SMP01810
                         RETURN (TAG2);                                           SMP01820
 801830   2  1     ELSE IF TAG2 <= -3 THEN                                        SMP01830
                      IF TAG1 = 0 THEN                                            SMP01840
                         RETURN (XTREE (LINK_PT).LINK);                           SMP01850
 801860   2  1        ELSE                                                        SMP01860
                         RETURN (TAG1);                                           SMP01870
 801880   2  1     IF TAG1 > 0 THEN                                               SMP01880
```

```
NUMBER LEV NT

                          XTREE (LINK_PT).LINK = TAG1;                    SMP01890
801900   2  1          IF TAG2 > 0 THEN                                    SMP01900
                          XTREE (LINK_PT + 1).LINK = TAG2;                SMP01910
                                                                          SMP01920
801920   2  1          END;                                                SMP01930
801930   2  0     RETURN (0);                                              SMP01940
                                                                          SMP01950
801950   2  0     END SEARCH;                                              SMP01960
                                                                          SMP01970
```

105

```
NUMBER LEV NT

801980   1  0  GETS: PROC (LIST,TERM_ITEM) RETURNS (CHAR (30) VAR);      SMP01980
                                                                          SMP01990
802000   2  0  DCL LIST CHAR (*) VAR,                                     SMP02000
                     TERM_ITEM CHAR (1),                                  SMP02010
                     RTN_LIST CHAR (30) VAR,                              SMP02020
                     I FIXED;                                             SMP02030
                                                                          SMP02040
802050   2  0  I = INDEX (LIST,TERM_ITEM);                                SMP02050
802060   2  0  IF I = 0 THEN DO;                                          SMP02060
802070   2  1     RTN_LIST = LIST;                                        SMP02070
802080   2  1     LIST = '';                                              SMP02080
802090   2  1     END;                                                    SMP02090
802100   2  0  ELSE DO;                                                   SMP02100
802110   2  1     RTN_LIST = SUBSTR (LIST,1,I - 1);                       SMP02110
802120   2  1     LIST = SUBSTR (LIST,I + 1);                             SMP02120
802130   2  1     END;                                                    SMP02130
802140   2  0  RETURN (RTN_LIST);                                         SMP02140
                                                                          SMP02150
802160   2  0  END GETS;                                                  SMP02160
                                                                          SMP02170
```

```
NUMBER LEV NT

802180   1  0  SETREL: PROC (LINK_PT,ATTR_LINK,CONST_LINK);                          SMP02180
                                                                                     SMP02190
802200   2  0  DCL (LINK_PT,ATTR_LINK,CONST_LINK,I) FIXED,                           SMP02200
                    CONSTTYP CHAR (30) VAR,                                          SMP02210
                    CONSTANT CHAR (80) VAR;                                          SMP02220
                                                                                     SMP02230
802240   2  0  NCOND = NCOND + 1;                                                    SMP02240
802250   2  0  ENTCOND (CUR_ES,NCOND).REL = XTREE (LINK_PT).LABEL;                   SMP02250
802260   2  0  ENTCOND (CUR_ES,NCOND).ATTRREF                                        SMP02260
                  = FIXED (GETS (XTREE (ATTR_LINK).LABEL,':'));                      SMP02270
802280   2  0  CONSTTYP = XTREE (CONST_LINK).LABEL;                                  SMP02280
802290   2  0  CONSTANT = GETS (CONSTTYP,':');                                       SMP02290
802300   2  0  IF CONSTTYP = 'MUL' THEN                                              SMP02300
                  CONSTANT = XCHNGE (FIXED (CONSTANT));                              SMP02310
802320   2  0  DO I = 1 TO 10 WHILE (CONSTANT ^= '');                                SMP02320
802330   2  1     ENTCOND (CUR_ES,NCOND).CDATA (I) = GETS (CONSTANT,',');            SMP02330
802340   2  1     END;                                                               SMP02340
                                                                                     SMP02350
802360   2  0  END SETREL;                                                           SMP02360
                                                                                     SMP02370
```

107

```
NUMBER LEV NT

802380   1  0  END SMPLFY;                                          SMP02380
```

```
 NUMBER LEV NT

     10       0  CNVERT: PROC (ENTITY,RETPTR);                                                  CNV00010
                                                                                                CNV00020
                 %INCLUDE ENTITY;***********************************************************CNV00030
 100010   1   0  DCL 1 ENTITY (0:12),                                                           ENT00010
                         2 NAME CHAR (30) VAR,                                                  ENT00020
                         2 DEPTH FIXED,                                                         ENT00030
                         2 VES_FN CHAR (2) VAR,                                                 ENT00040
                         2 WHERE FIXED,                                                         ENT00050
                         2 N_PARENT (2) FIXED,                                                  ENT00060
                         2 VES_PAR PTR,                                                         ENT00070
                         2 ATTR (15),                                                           ENT00080
                            3 VES_KEY BIT (1),                                                  ENT00090
                            3 CART_KEY BIT (1),                                                 ENT00100
                            3 SING_OCC BIT (1),                                                 ENT00110
                            3 A_PARENT FIXED,                                                   ENT00120
                            3 USES CHAR (30) VAR,                                               ENT00130
                            3 LIST PTR;                                                         ENT00140
 100150   1   0  DCL 1 N_MAP (2) BASED,                                                         ENT00150
                         2 NUM (15) FIXED;                                                      ENT00160
                 ****************                                                               CNV00030
                 %INCLUDE ATTRIB;***********************************************************CNV00040
 300010   1   0  DCL 1 ATTRIB BASED,                                                            ATT00010
                      2 LEVEL FIXED,                                                            ATT00020
                      2 ITEM CHAR (50) VAR,                                                     ATT00030
                      2 NEXT PTR;                                                               ATT00040
                 ****************                                                               CNV00040
                 %INCLUDE ARETE;************************************************************CNV00050
 500020   1   0  DCL 1 RETE_ARG BASED (RETEP),  /* WILL ALSO BE RETE_RTN */                     F2A00020
                        2 CTL_INFO,                                                             F2A00030
                          3 LEN FIXED BIN (15),                                                 F2A00040
                          3 CBTP FIXED BIN (15) INIT (21),                                      F2A00050
                          3 PTR PTR,                                                            F2A00060
                        2 NUM_ATTR FIXED,                                                       F2A00070
                        2 NUM_COND FIXED,    /* NUMBER OF CONDITIONS */                         F2A00080
                        2 ENT,                                                                  F2A00090
                          3 NAME CHAR (30) VAR,  /* ENTITY SET NAME */                          F2A00100
                          3 DEPTH FIXED,     /* FILLED IN WHEN RETURNED */                      F2A00110
                          3 ATTR (NATTR REFER (RETE_ARG.NUM_ATTR)),                             F2A00120
                            4 SING_OCC BIT (1), /* IF SINGLE OCCUR THEN SET */                  F2A00130
                            4 A_PARENT FIXED, /* PARENT NUMBER */                               F2A00140
                            4 USES CHAR(30) VAR, /* ATTRIBUTE NAME */                           F2A00150
                            4 LIST PTR, /* POINT TO LIST OF OCC OF THIS ATTR IF ANY */          F2A00160
                        2 COND (NCOND REFER (RETE_ARG.NUM_COND)),                               F2A00170
                          3 ATTRREF FIXED, /* POINT TO ATTR IN ATTR ARRAY ABOVE */              F2A00180
                          3 NEG BIT(1), /* NEGATION OF RELATOR */                               F2A00190
                          3 REL CHAR(1), /* '=', '<', '>' */                                    F2A00200
                          3 CDATA (10) CHAR (50) VAR,  /* UP TO 10 'MULTI' ITMES */             F2A00210
                        2 RTN_CODE FIXED BIN,                                                   F2A00220
                        2 NEXT_PTR PTR;                                                         F2A00230
```

```
NUMBER LEV NT

500240   1  0  DCL RETEP PTR;                                                      F2A00240
                                                                                   F2A00250
500260   1  0  DCL 1 RETE_RTN1 BASED,          /* USED WHEN RETURNED */           F2A00260
                     2 CTL,                                                        F2A00270
                       3 LEN FIXED BIN(31),                                        F2A00280
                       3 CBTP FIXED BIN (31) INIT (46),                            F2A00290
                       3 PTR PTR,                                                  F2A00300
                     2 LEVEL FIXED BIN, /* OCCUR NUMBER */                         F2A00310
                   . 2 ITEM CHAR (50),                                             F2A00320
                     2 NEXT_PTR PTR; /* POINT TO NEXT ON THE SAME ATTRIBUTE LIST */ F2A00330
               ***************                                                     CNV00050
                                                                                   CNV00060
600070   1  0  DCL (RETPTR,TAILR,P,TAILP) PTR,                                     CNV00070
                   (CUR_ES,I) FIXED;                                               CNV00080
                                                                                   CNV00090
600100   1  0  TAILR = RETPTR;                                                     CNV00100
600110   1  0  DO CUR_ES = 1 TO 12 WHILE (ENTITY (CUR_ES).VES_FN ^= '');           CNV00110
600120   1  1     IF ENTITY (CUR_ES).VES_FN = 'R' THEN DO;                         CNV00120
600130   1  2        ENTITY (CUR_ES).DEPTH =  TAILR -> RETE_ARG.ENT.DEPTH;         CNV00130
600140   1  2        DO I = 1 TO 15 WHILE (ENTITY (CUR_ES).ATTR (I).USES ^= '');   CNV00140
600150   1  3           ENTITY (CUR_ES).ATTR (I).SING_OCC =                        CNV00150
                           TAILR -> RETE_ARG.ENT.ATTR (I).SING_OCC;                CNV00160
600170   1  3           Q = TAILR -> RETE_ARG.ENT.ATTR (I).LIST;                   CNV00170
600180   1  3           DO WHILE (Q ^= NULL ());                                   CNV00180
600190   1  4              ALLOCATE ATTRIB SET (P);                                CNV00190
600200   1  4              P -> ATTRIB.ITEM = Q -> RETE_RTN1.ITEM;                 CNV00200
600210   1  4              P -> ATTRIB.LEVEL = Q -> RETE_RTN1.LEVEL;               CNV00210
600220   1  4              IF ENTITY (CUR_ES).ATTR (I).LIST = NULL () THEN         CNV00220
                              ENTITY (CUR_ES).ATTR (I).LIST = P;                   CNV00230
600240   1  4              ELSE                                                    CNV00240
                              TAILP -> ATTRIB.NEXT = P;                            CNV00250
600260   1  4              TAILP = P;                                              CNV00260
600270   1  4              Q = Q-> RETE_RTN1.NEXT_PTR;                             CNV00270
600280   1  4              END;                                                    CNV00280
600290   1  3           TAILP -> ATTRIB.NEXT = NULL ();                            CNV00290
600300   1  3           END;                                                       CNV00300
600310   1  2        TAILR = TAILR -> RETE_ARG.NEXT_PTR;                           CNV00310
600320   1  2        END;                                                          CNV00320
600330   1  1     END;                                                             CNV00330
                                                                                   CNV00340
600350   1  0  END CNVERT;                                                         CNV00350
```

```
 NUMBER LEV NT

     10       0  XECUTE: PROC (XTREE,ENTITY,XCHNGE);                                    XEC00010
                                                                                        XEC00020
                                                                                        XEC00030
                                                                                        XEC00040
                 /*******************************************************************/ XEC00050
                 /*                                                                 */ XEC00060
                 /*EACH NODE IN THIS TREE TRAVERSAL PROGRAM HAS A UNIT_VERT TAG     */ XEC00070
                 /*AND THESE TAGS ARE PROPAGATED UP THE TREE FROM CHILDREN TO       */ XEC00080
                 /*PARENTS.  PROPAGATION OF THE TAGS FOLLOWS THE RULE THAT UNITARY  */ XEC00090
                 /*ITEMS CAN ONLY RESULT FROM THE GENERIC OPERATION BETWEEN TWO     */ XEC00100
                 /*UNITARY ITEMS OR FROM A BUILTIN FUNCTION WHICH RETURNS A UNITARY */ XEC00110
                 /*ITEM; IN ALL OTHER CASES THE RETURNED ITEMS ARE OF TYPE VERTICAL.*/ XEC00120
                 /*THIS SCHEME ALLOWS GENERIC OPERATORS TO DESIGNATE THE EXPECTED   */ XEC00130
                 /*TYPING OF THEIR RETURN ITEMS.                                    */ XEC00140
                 /*                                                                 */ XEC00150
                 /*******************************************************************/ XEC00160
                                                                                        XEC00170
                 %INCLUDE XTREE;*******************************************************XEC00180
 100010    1  0  DCL 1 XTREE (1000),                                                    XTR00010
                     2 LABEL CHAR (30) VAR,                                             XTR00020
                     2 CHILD FIXED,                                                     XTR00030
                     2 LINK FIXED;                                                      XTR00040
                 ***************                                                        XEC00180
                 %INCLUDE ENTITY;******************************************************XEC00190
 200010    1  0  DCL 1 ENTITY (0:12),                                                   ENT00010
                        2 NAME CHAR (30) VAR,                                           ENT00020
                        2 DEPTH FIXED,                                                  ENT00030
                        2 VES_FN CHAR (2) VAR,                                          ENT00040
                        2 WHERE FIXED,                                                  ENT00050
                        2 N_PARENT (2) FIXED,                                           ENT00060
                        2 VES_PAR PTR,                                                  ENT00070
                        2 ATTR (15),                                                    ENT00080
                           3 VES_KEY BIT (1),                                           ENT00090
                           3 CART_KEY BIT (1),                                          ENT00100
                           3 SING_OCC BIT (1),                                          ENT00110
                           3 A_PARENT FIXED,                                            ENT00120
                           3 USES CHAR (30) VAR,                                        ENT00130
                           3 LIST PTR;                                                  ENT00140
 200150    1  0  DCL 1 N_MAP (2) BASED,                                                 ENT00150
                        2 NUM (15) FIXED;                                               ENT00160
                 ****************                                                       XEC00190
                 %INCLUDE XCHNGE;******************************************************XEC00200
 300010    1  0  DCL XCHNGE (20) CHAR (80) VAR;                                         IND00010
                 ****************                                                       XEC00200
                 %INCLUDE ATTRIB;******************************************************XEC00210
 400010    1  0  DCL 1 ATTRIB BASED,                                                    ATT00010
                     2 LEVEL FIXED,                                                     ATT00020
                     2 ITEM CHAR (50) VAR,                                              ATT00030
                     2 NEXT PTR;                                                        ATT00040
```

```
NUMBER LEV NT

                 ****************

400230    1  0  DCL UNIT_VERT BIT (1),
                    (CUR_ES,M,COPY_COUNT,KEYSET (15),NEWLVL (O:2,15)) FIXED,
                    (SLCTLIST,ATAIL (O:2,15)) PTR;

400270    1  0  DO CUR_ES = 1 TO 12 WHILE (ENTITY (CUR_ES).VES_FN ^= '');
400280    1  1     SELECT (ENTITY (CUR_ES).VES_FN);
400290    1  2        WHEN ('S')
                         CALL COPYALL (ENTITY (CUR_ES).N_PARENT (1));
400310    1  2        WHEN ('MU')
                         CALL MUNION;
400330    1  2        WHEN ('MI')
                         CALL MINTER;
400350    1  2        WHEN ('SU') DO;
400360    1  3           CALL MUNION;
400370    1  3           CALL SINGLE;
400380    1  3           END;
400390    1  2        WHEN ('SI') DO;
400400    1  3           CALL MINTER;
400410    1  3           CALL SINGLE;
400420    1  3           END;
400430    1  2        WHEN ('CS')
                         CALL CRTESN;
400450    1  2        OTHERWISE;
400460    1  2        END;
400470    1  1     IF ENTITY (CUR_ES).WHERE > O THEN
                      IF OPERATE (SLCTLIST,ENTITY (CUR_ES).WHERE) THEN
                         CALL DECRSET;
400500    1  1        ELSE IF SLCTLIST -> ATTRIB.ITEM = ':FALSE' THEN DO;
400510    1  2           DO M = 1 TO 15 WHILE (ENTITY (CUR_ES).ATTR (M).USES ^= '');
400520    1  3              CALL DISPOSE (ENTITY (CUR_ES).ATTR (M).LIST);
400530    1  3              END;
400540    1  2           ENTITY (CUR_ES).DEPTH = O;
400550    1  2           END;
400560    1  1     END;
```

```
NUMBER LEV NT

400580   1  0  OPERATE: PROC (LIST1,LINK_PT) RETURNS (BIT (1)) RECURSIVE;               XEC00580
                                                                                        XEC00590
               /*MODIFIES LIST1*/                                                       XEC00600
400610   2  0  DCL (LIST1,LIST2) PTR,                                                   XEC00610
                    LINK_PT FIXED,                                                      XEC00620
                    (TERM_TYP,TERM_NM) CHAR (30) VAR,                                   XEC00630
                    (UNIT_VERT1,UNIT_VERT2) BIT (1);                                    XEC00640
                                                                                        XEC00650
400660   2  0  IF XTREE (LINK_PT).CHILD = 0 THEN DO;                                    XEC00660
400670   2  1     TERM_TYP = XTREE (LINK_PT).LABEL;                                     XEC00670
400680   2  1     TERM_NM = GETS (TERM_TYP,':');                                        XEC00680
400690   2  1     SELECT (TERM_TYP);                                                    XEC00690
400700   2  2        WHEN ('NTH')                                                       XEC00700
                        UNIT_VERT1 = CREATE_LIST (LIST1,FIXED (TERM_NM));               XEC00710
400720   2  2        WHEN ('A','S')                                                     XEC00720
                        UNIT_VERT1 = CREATE_CONST (LIST1,TERM_NM);                      XEC00730
400740   2  2        WHEN ('MUL')                                                       XEC00740
                        CALL CREATE_CONST (LIST1,':' || TERM_NM);                       XEC00750
400760   2  2        OTHERWISE                                                          XEC00760
                        UNIT_VERT1 = GO_DOWNO (LIST1,TERM_NM);                          XEC00770
400780   2  2        END;                                                               XEC00780
400790   2  1     END;                                                                  XEC00790
400800   2  0  ELSE IF XTREE (LINK_PT).CHILD = 1 THEN DO;                               XEC00800
400810   2  1     UNIT_VERT1 = OPERATE (LIST1,XTREE (LINK_PT).LINK);                    XEC00810
400820   2  1     UNIT_VERT1 = GO_DOWN1 (LIST1,UNIT_VERT1,XTREE (LINK_PT).LABEL);       XEC00820
400830   2  1     END;                                                                  XEC00830
400840   2  0  ELSE IF XTREE (LINK_PT).CHILD = 2 THEN DO;                               XEC00840
400850   2  1     UNIT_VERT1 = OPERATE (LIST1,XTREE (LINK_PT).LINK);                    XEC00850
400860   2  1     UNIT_VERT2 = OPERATE (LIST2,XTREE (LINK_PT + 1).LINK);                XEC00860
400870   2  1     IF SUBSTR (LIST2 -> ATTRIB.ITEM,1,1) = ':' THEN                       XEC00870
                     UNIT_VERT1 =                                                       XEC00880
                        MULTEQ2 (LIST1,                                                 XEC00890
                              XCHNGE (FIXED (SUBSTR (LIST2 -> ATTRIB.ITEM,2))));        XEC00900
400910   2  1     ELSE                                                                  XEC00910
                     UNIT_VERT1 = GO_DOWN2 (LIST1,LIST2,UNIT_VERT1,UNIT_VERT2,          XEC00920
                                     XTREE (LINK_PT).LABEL);                            XEC00930
400940   2  1     END;                                                                  XEC00940
400950   2  0  ELSE IF XTREE (LINK_PT).LABEL = 'STR' THEN                               XEC00950
                  UNIT_VERT1 = STR (LIST1,LINK_PT);                                     XEC00960
400970   2  0  ELSE                                                                     XEC00970
                  /*BAD CHILDREN*/;                                                     XEC00980
400990   2  0  RETURN (UNIT_VERT1);                                                     XEC00990
                                                                                        XEC01000
401010   2  0  END OPERATE;                                                             XEC01010
                                                                                        XEC01020
```

```
NUMBER LEV NT

401030   1  0  STR: PROC (LIST,LINK_PT) RETURNS (BIT (1)) RECURSIVE;                XECO1030
                                                                                    XECO1040
401050   2  0  DCL (LIST,LISTP,P (4)) PTR,                                          XECO1050
                   (LINK_PT,START_PT,END_PT) FIXED,                                 XECO1060
                   (UNIT_VERT,ENDING,RELATIVE,FIX_START,FIX_END) BIT (1) INIT ('O'B), XECO1070
                   UV (4) BIT (1) INIT ((4)'O'B),                                   XECO1080
                   STRING CHAR (30) VAR;                                            XECO1090
                                                                                    XECO1100
401110   2  0  UNIT_VERT = OPERATE (LIST,XTREE (LINK_PT).LINK);                     XECO1110
401120   2  0  UV (1) = OPERATE (P (1),XTREE (LINK_PT + 1).LINK);                   XECO1120
401130   2  0  IF XTREE (LINK_PT + 2).LINK = -1 THEN                                XECO1130
                  FIX_START = '1'B;                                                 XECO1140
401150   2  0  ELSE                                                                 XECO1150
                  UV (2) = OPERATE (P (2),XTREE (LINK_PT + 2).LINK);                XECO1160
401170   2  0  ENDING = (XTREE (LINK_PT + 3).LINK = -1);                            XECO1170
401180   2  0  IF ^ENDING THEN DO;                                                  XECO1180
401190   2  1     RELATIVE = (XTREE (LINK_PT + 3).LINK = -3);                       XECO1190
401200   2  1     UV (3) = OPERATE (P (3),XTREE (LINK_PT + 4).LINK);                XECO1200
401210   2  1     IF XTREE (LINK_PT + 5).LINK = -1 THEN                             XECO1210
                     FIX_END = '1'B;                                                XECO1220
401230   2  1     ELSE                                                              XECO1230
                     UV (4) = OPERATE (P (4),XTREE (LINK_PT + 5).LINK);             XECO1240
401250   2  1     END;                                                              XECO1250
401260   2  0  LISTP = LIST;                                                        XECO1260
401270   2  0  DO WHILE (LISTP ^= NULL ());                                         XECO1270
401280   2  1     STRING = LISTP -> ATTRIB.ITEM;                                    XECO1280
401290   2  1     IF FIX_START THEN                                                 XECO1290
                     START_PT = FIND_PT (1,0,'O'B);                                 XECO1300
401310   2  1     ELSE                                                              XECO1310
                     START_PT = FIND_PT (1,1,'1'B);                                 XECO1320
401330   2  1     START_PT = MIN (MAX (START_PT,1),LENGTH (STRING) + 1);            XECO1330
401340   2  1     IF ENDING THEN                                                    XECO1340
                     LISTP -> ATTRIB.ITEM = SUBSTR (STRING,START_PT);               XECO1350
401360   2  1     ELSE DO;                                                          XECO1360
401370   2  2        IF FIX_END THEN                                                XECO1370
                        END_PT = FIND_PT (3,0,'O'B);                                XECO1380
401390   2  2        ELSE IF RELATIVE THEN                                          XECO1390
                        END_PT = FIND_PT (3,START_PT,'O'B);                         XECO1400
401410   2  2        ELSE                                                           XECO1410
                        END_PT = FIND_PT (3,1,'O'B);                                XECO1420
401430   2  2        END_PT = MIN (END_PT,LENGTH (STRING));                         XECO1430
401440   2  2        LISTP -> ATTRIB.ITEM =                                         XECO1440
                        SUBSTR (STRING,                                             XECO1450
                                START_PT,                                           XECO1460
                                MAX (END_PT - START_PT + 1,0));                     XECO1470
401480   2  2        END;                                                           XECO1480
401490   2  1     DO I = 1 TO 4;                                                    XECO1490
401500   2  2        IF UV (I) THEN                                                 XECO1500
                        P (I) = P (I) -> ATTRIB.NEXT;                               XECO1510
```

```
 NUMBER LEV NT
                                                                        XECO1520
 401520   2  2         END;                                             XECO1530
 401530   2  1      LISTP = LISTP -> ATTRIB.NEXT;                       XECO1540
 401540   2  1      END;                                                XECO1550
 401550   2  0  DO I = 1 TO 4;                                          XECO1560
 401560   2  1      CALL DISPOSE (P (I));                               XECO1570
 401570   2  1      END;                                                XECO1580
 401580   2  0  RETURN (UNIT_VERT);                                     XECO1590
```

```
NUMBER LEV NT

401600   2  0  FIND_PT: PROC (PNUM,FND_LOC,OFFSET) RETURNS (FIXED);              XECO1600
                                                                                  XECO1610
401620   3  0  DCL (PNUM,FND_LOC,NUM_OCC,I) FIXED,                               XECO1620
                    OFFSET BIT (1),                                               XECO1630
                    TEMPSTR CHAR (30) VAR;                                        XECO1640
                                                                                  XECO1650
401660   3  0  ON CONVERSION GOTO MESS;                                           XECO1660
401670   3  0  NUM_OCC = FIXED (P (PNUM) -> ATTRIB.ITEM);                         XECO1670
401680   3  0  IF FND_LOC < 1 THEN                                                XECO1680
                  RETURN (NUM_OCC);                                               XECO1690
401700   3  0  TEMPSTR = P (PNUM + 1) -> ATTRIB.ITEM || '|';                      XECO1700
401710   3  0  DO I = FND_LOC TO LENGTH (STRING) WHILE (NUM_OCC > 0);             XECO1710
401720   3  1     IF TMPLTE (SUBSTR (STRING,I),TEMPSTR) THEN                      XECO1720
                     NUM_OCC = NUM_OCC -1;                                        XECO1730
401740   3  1     END;                                                            XECO1740
401750   3  0  IF OFFSET THEN                                                     XECO1750
                  I = I + LENGTH (TEMPSTR);                                       XECO1760
401770   3  0  RETURN (I - 2);                                                    XECO1770
                                                                                  XECO1780
401790   3  0  MESS:                                                              XECO1790
               IF OFFSET THEN                                                     XECO1800
                  NUM_OCC = LENGTH (STRING) + 1;                                  XECO1810
401820   3  0  ELSE                                                               XECO1820
                  NUM_OCC = 0;                                                    XECO1830
401840   3  0  PUT FILE (ERROR) SKIP EDIT                                         XECO1840
                  ('CONV ERROR FOR STR_OP',                                       XECO1850
                   'STRING ARGUMENT =$',STRING,'$$',                              XECO1860
                   'NUM_OCC ARGUMENT =$',P (PNUM) -> ATTRIB.ITEM,'$$',            XECO1870
                   'FORCED NUM_OCC =$',CHAR (NUM_OCC),'$$')                       XECO1880
                  (A,3 (SKIP,A,A,A));                                             XECO1890
401900   3  0  RETURN (NUM_OCC);                                                  XECO1900
                                                                                  XECO1910
401920   3  0  END FIND_PT;                                                       XECO1920
                                                                                  XECO1930
401940   2  0  END STR;                                                           XECO1940
                                                                                  XECO1950
```

```
NUMBER LEV NT

401960   1  0  GO_DOWNO: PROC (LIST,OPNM) RETURNS (BIT (1));        XEC01960
                                                                    XEC01970
               /*MODIFIES LIST*/                                    XEC01980
401990   2  0  DCL LIST PTR,                                        XEC01990
                   OPNM CHAR (30) VAR;                              XEC02000
402010   2  0  DCL DATE BUILTIN;                                    XEC02010
                                                                    XEC02020
402030   2  0  ALLOCATE ATTRIB SET (LIST);                          XEC02030
402040   2  0  LIST -> ATTRIB.NEXT = NULL ();                       XEC02040
402050   2  0  SELECT (OPNM);                                       XEC02050
402060   2  1     WHEN ('DATE')                                     XEC02060
                     LIST -> ATTRIB.ITEM = DATE;                    XEC02070
402080   2  1     OTHERWISE;                                        XEC02080
402090   2  1     END;                                              XEC02090
402100   2  0  RETURN ('0'B);                                       XEC02100
                                                                    XEC02110
402120   2  0  END GO_DOWNO;                                        XEC02120
                                                                    XEC02130
```

```
 NUMBER LEV NT

 402140   1  0  GO_DOWN1: PROC (LIST,UNIT_VERT,REL) RETURNS (BIT (1));                XECO2140
                                                                                     XECO2150
                /*MODIFIES LIST,UNIT_VERT*/                                          XECO2160
 402170   2  0  DCL (LIST,P) PTR,                                                    XECO2170
                    UNIT_VERT BIT (1),                                               XECO2180
                    REL CHAR (30) VAR;                                               XECO2190
                                                                                     XECO2200
 402210   2  0  IF ^RET_UNITARY (LIST,REL,UNIT_VERT) THEN                            XECO2210
                   IF UNIT_VERT THEN                                                 XECO2220
                      LIST -> ATTRIB.ITEM = UNARY (LIST -> ATTRIB.ITEM,REL);         XECO2230
 402240   2  0     ELSE DO;                                                          XECO2240
 402250   2  1        P = LIST;                                                      XECO2250
 402260   2  1        DO WHILE (P ^= NULL ());                                       XECO2260
 402270   2  2           P -> ATTRIB.ITEM = UNARY (P -> ATTRIB.ITEM,REL);            XECO2270
 402280   2  2           P = P -> ATTRIB.NEXT;                                       XECO2280
 402290   2  2           END;                                                        XECO2290
 402300   2  1        END;                                                           XECO2300
 402310   2  0  RETURN (UNIT_VERT);                                                  XECO2310
                                                                                     XECO2320
 402330   2  0  END GO_DOWN1;                                                        XECO2330
                                                                                     XECO2340
```

```
 NUMBER LEV NT

 402350   1  0  RET_UNITARY: PROC (LIST,REL,UNIT_VERT) RETURNS (BIT (1));                 XECO2350
                                                                                          XECO2360
                /*MODIFIES? LIST,UNIT_VERT*/                                              XECO2370
 402380   2  0  DCL (LIST,P) PTR,                                                         XECO2380
                     REL CHAR (30) VAR,                                                   XECO2390
                     UNIT_VERT BIT (1),                                                   XECO2400
                     RESULT FIXED;                                                        XECO2410
                                                                                          XECO2420
 402430   2  0  ON CONVERSION GOTO MESS;                                                  XECO2430
 402440   2  0  ON OVERFLOW GOTO MESS;                                                    XECO2440
 402450   2  0  IF INDEX ('DEPTH,MAX,MIN,SUM',REL) ^= 0 THEN DO;                          XECO2450
 402460   2  1     IF REL = 'DEPTH' THEN DO;                                              XECO2460
 402470   2  2        LIST -> ATTRIB.ITEM = ENTITY (CUR_ES).DEPTH;                        XECO2470
 402480   2  2        END;                                                                XECO2480
 402490   2  1     ELSE DO;                                                               XECO2490
 402500   2  2        RESULT = FIXED (LIST -> ATTRIB.ITEM);                               XECO2500
 402510   2  2        P = LIST -> ATTRIB.NEXT;                                            XECO2510
 402520   2  2        DO WHILE (P ^= NULL ());                                            XECO2520
 402530   2  3           SELECT (REL);                                                    XECO2530
 402540   2  4              WHEN ('MAX')                                                  XECO2540
                               IF RESULT < FIXED (P -> ATTRIB.ITEM) THEN                  XECO2550
                                  RESULT = FIXED (P -> ATTRIB.ITEM);                      XECO2560
 402570   2  4              WHEN ('MIN')                                                  XECO2570
                               IF RESULT > FIXED (P -> ATTRIB.ITEM) THEN                  XECO2580
                                  RESULT = FIXED (P -> ATTRIB.ITEM);                      XECO2590
 402600   2  4              WHEN ('SUM')                                                  XECO2600
                               RESULT = RESULT + FIXED (P -> ATTRIB.ITEM);                XECO2610
 402620   2  4              OTHERWISE;                                                    XECO2620
 402630   2  4              END;                                                          XECO2630
 402640   2  3           P = P -> ATTRIB.NEXT;                                            XECO2640
 402650   2  3           END;                                                             XECO2650
 402660   2  2        LIST -> ATTRIB.ITEM = FIXED (RESULT);                               XECO2660
 402670   2  2        END;                                                                XECO2670
 402680   2  1     CALL DISPOSE (LIST -> ATTRIB.NEXT);                                    XECO2680
 402690   2  1     UNIT_VERT = '0'B;                                                      XECO2690
 402700   2  1     RETURN ('1'B);                                                         XECO2700
 402710   2  1     END;                                                                   XECO2710
 402720   2  0  ELSE                                                                      XECO2720
                   RETURN ('0'B);                                                         XECO2730
                                                                                          XECO2740
 402750   2  0  MESS:                                                                     XECO2750
                PUT FILE (ERROR) SKIP EDIT                                                XECO2760
                   ('CONV/OVFLOW/BAD_ARG ERROR FOR RET_UNITARY_OP',                       XECO2770
                    'ITEM ARGUMENT =$',P -> ATTRIB.ITEM,'$$',                             XECO2780
                    'UNITARY UNIARY OPERATOR =$',REL,'$$',                                XECO2790
                    'FORCED RESULT =$:EMPTY$$')                                           XECO2800
                   (A,2 (SKIP,A,A,A),SKIP,A);                                             XECO2810
 402820   2  0  LIST -> ATTRIB.ITEM = ':EMPTY';                                           XECO2820
 402830   2  0  LIST -> ATTRIB.NEXT = NULL ();                                            XECO2830
```

119

```
NUMBER LEV NT

402840   2  0  UNIT_VERT = '0'B;                          XECO2840
402850   2  0  RETURN ('1'B);                             XECO2850
                                                          XECO2860
402870   2  0  END RET_UNITARY;                           XECO2870
                                                          XECO2880
```

```
NUMBER LEV NT

402890   1  0  UNARY: PROC (ITEM,REL) RETURNS (CHAR (50) VAR);                  XEC02890
                                                                                XEC02900
402910   2  0  DCL ITEM CHAR (50) VAR,                                          XEC02910
                    REL CHAR (30) VAR;                                          XEC02920
                                                                                XEC02930
402940   2  0  ON CONVERSION GOTO MESS;                                         XEC02940
402950   2  0  ON OVERFLOW GOTO MESS;                                           XEC02950
402960   2  0  SELECT (REL);                                                    XEC02960
402970   2  1     WHEN ('^')                                                    XEC02970
                     IF ITEM = ':FALSE' THEN                                    XEC02980
                        RETURN (':TRUE');                                       XEC02990
403000   2  1        ELSE IF ITEM = ':TRUE' THEN                                XEC03000
                        RETURN (':FALSE');                                      XEC03010
403020   2  1     WHEN ('ZER')                                                  XEC03020
                     IF FIXED (ITEM) = 0 THEN                                   XEC03030
                        RETURN (CHAR (1));                                      XEC03040
403050   2  1        ELSE                                                       XEC03050
                        RETURN (CHAR (0));                                      XEC03060
403070   2  1     WHEN ('POS')                                                  XEC03070
                     IF FIXED (ITEM) > 0 THEN                                   XEC03080
                        RETURN (CHAR (1));                                      XEC03090
403100   2  1        ELSE                                                       XEC03100
                        RETURN (CHAR (0));                                      XEC03110
403120   2  1     WHEN ('-P')                                                   XEC03120
                     RETURN (CHAR (-1 * FIXED (ITEM)));                         XEC03130
403140   2  1     WHEN ('SGN')                                                  XEC03140
                     RETURN (CHAR (SIGN (FIXED (ITEM))));                       XEC03150
403160   2  1     WHEN ('ABS')                                                  XEC03160
                     RETURN (CHAR (ABS (FIXED (ITEM))));                        XEC03170
403180   2  1     OTHERWISE;                                                    XEC03180
403190   2  1     END;                                                          XEC03190
                                                                                XEC03200
403210   2  0  MESS:                                                            XEC03210
               PUT FILE (ERROR) SKIP EDIT                                       XEC03220
                  ('CONV/OVFLOW/BAD_ARG ERROR FOR UNARY_OP',                    XEC03230
                   'ITEM ARGUMENT =$',ITEM,'$$',                                XEC03240
                   'MIXED UNARY OPERATOR =$',REL,'$$',                          XEC03250
                   'FORCED RESULT =$:EMPTY$$')                                  XEC03260
                  (A,2 (SKIP,A,A,A),SKIP,A);                                    XEC03270
403280   2  0  RETURN (':EMPTY');                                               XEC03280
                                                                                XEC03290
403300   2  0  END UNARY;                                                       XEC03300
                                                                                XEC03310
```

```
NUMBER LEV NT

403320   1  0  MULTEQ2: PROC (LIST,MULT_ITEMS);                                   XECO3320
                                                                                  XECO3330
               /*MODIFIES LIST*/                                                  XECO3340
403350   2  0  DCL (LIST,P) PTR,                                                  XECO3350
                   (MULT_ITEMS,MULT_ITEMS1) CHAR (80) VAR,                        XECO3360
                   STOP BIT (1);                                                  XECO3370
                                                                                  XECO3380
403390   2  0  P = LIST;                                                          XECO3390
403400   2  0  DO WHILE (P ^= NULL ());                                           XECO3400
403410   2  1     MULT_ITEMS1 = MULT_ITEMS;                                       XECO3410
403420   2  1     STOP = '0'B;                                                    XECO3420
403430   2  1     DO WHILE (^STOP & MULT_ITEMS1 ^= '');                           XECO3430
403440   2  2        STOP = TMPLTE (P -> ATTRIB.ITEM,GETS (MULT_ITEMS1,','));     XECO3440
403450   2  2        END;                                                         XECO3450
403460   2  1     IF STOP THEN                                                    XECO3460
                     P -> ATTRIB.ITEM = ':TRUE';                                  XECO3470
403480   2  1     ELSE                                                            XECO3480
                     P -> ATTRIB.ITEM = ':FALSE';                                 XECO3490
403500   2  1     P = P -> ATTRIB.NEXT;                                           XECO3500
403510   2  1     END;                                                            XECO3510
                                                                                  XECO3520
403530   2  0  END MULTEQ2;                                                       XECO3530
                                                                                  XECO3540
```

```
NUMBER LEV NT

403550   1  0  GO_DOWN2: PROC (LIST1,LIST2,UNIT_VERT1,UNIT_VERT2,REL)                XEC03550
                          RETURNS (BIT (1));                                          XEC03560
                                                                                      XEC03570
               /*MODIFIES LIST1 OR LIST2*/                                            XEC03580
403590   2  0  DCL (LIST1,LIST2,P (2)) PTR,                                           XEC03590
                   (UNIT_VERT1,UNIT_VERT2,UNIT_VERT,ONE_VERT) BIT (1),                XEC03600
                   REL CHAR (30) VAR,                                                 XEC03610
                   STORE_NUM FIXED;                                                   XEC03620
                                                                                      XEC03630
403640   2  0  UNIT_VERT = UNIT_VERT1 | UNIT_VERT2;                                   XEC03640
403650   2  0  IF ^UNIT_VERT THEN                                                     XEC03650
                  LIST1 -> ATTRIB.ITEM = BINARY (LIST1 -> ATTRIB.ITEM,                XEC03660
                                                 LIST2 -> ATTRIB.ITEM,REL);           XEC03670
403680   2  0  ELSE DO;                                                               XEC03680
403690   2  1     ONE_VERT = ^(UNIT_VERT1 & UNIT_VERT2);                              XEC03690
403700   2  1     STORE_NUM = 1;                                                      XEC03700
403710   2  1     IF ^UNIT_VERT1 THEN                                                 XEC03710
                     STORE_NUM = 2;                                                   XEC03720
403730   2  1     P (1) = LIST1;                                                      XEC03730
403740   2  1     P (2) = LIST2;                                                      XEC03740
403750   2  1     DO WHILE (P (STORE_NUM) ^= NULL ());                                XEC03750
403760   2  2        P (STORE_NUM) -> ATTRIB.ITEM =                                   XEC03760
                        BINARY (P (1) -> ATTRIB.ITEM,                                 XEC03770
                                P (2) -> ATTRIB.ITEM,REL);                            XEC03780
403790   2  2        IF ONE_VERT THEN                                                 XEC03790
                        P (STORE_NUM) = P (STORE_NUM) -> ATTRIB.NEXT;                 XEC03800
403810   2  2        ELSE DO;                                                         XEC03810
403820   2  3           P (1) = P (1) -> ATTRIB.NEXT;                                 XEC03820
403830   2  3           P (2) = P (2) -> ATTRIB.NEXT;                                 XEC03830
403840   2  3           END;                                                          XEC03840
403850   2  2        END;                                                             XEC03850
403860   2  1     END;                                                                XEC03860
403870   2  0  LIST1 = P (STORE_NUM);                                                 XEC03870
403880   2  0  LIST2 = NULL ();                                                       XEC03880
403890   2  0  CALL DISPOSE (P (3 - STORE_NUM));                                      XEC03890
403900   2  0  RETURN (UNIT_VERT);                                                    XEC03900
                                                                                      XEC03910
403920   2  0  END GO_DOWN2;                                                          XEC03920
                                                                                      XEC03930
```

```
NUMBER LEV NT

403940    1  0  BINARY: PROC (ITEM1,ITEM2,REL) RETURNS (CHAR (50) VAR);          XEC03940
                                                                                  XEC03950
403960    2  0  DCL (ITEM1,ITEM2) CHAR (50) VAR,                                 XEC03960
                     REL CHAR (30) VAR;                                          XEC03970
                                                                                  XEC03980
403990    2  0  ON CONVERSION GOTO MESS;                                         XEC03990
404000    2  0  ON OVERFLOW GOTO MESS;                                           XEC04000
404010    2  0  ON ZERODIVIDE GOTO MESS;                                         XEC04010
```

```
NUMBER LEV NT

404020   2  0  SELECT (REL);                                                     XECO4020
404030   2  1     WHEN ('=')                                                     XECO4030
                     IF ITEM1 = ITEM2                                            XECO4040
                        ¦ TMPLTE (ITEM1,ITEM2)                                   XECO4050
                        ¦ TMPLTE (ITEM2,ITEM1) THEN                              XECO4060
                        RETURN (':TRUE');                                        XECO4070
404080   2  1        ELSE                                                        XECO4080
                        RETURN (':FALSE');                                       XECO4090
404100   2  1     WHEN ('>')                                                     XECO4100
                     IF ITEM1 > ITEM2 THEN                                       XECO4110
                        RETURN (':TRUE');                                        XECO4120
404130   2  1        ELSE                                                        XECO4130
                        RETURN (':FALSE');                                       XECO4140
404150   2  1     WHEN ('<')                                                     XECO4150
                     IF ITEM1 < ITEM2 THEN                                       XECO4160
                        RETURN (':TRUE');                                        XECO4170
404180   2  1        ELSE                                                        XECO4180
                        RETURN (':FALSE');                                       XECO4190
404200   2  1     WHEN ('AND')                                                   XECO4200
                     IF ITEM1 = ':TRUE' & ITEM2 = ':TRUE' THEN                   XECO4210
                        RETURN (':TRUE');                                        XECO4220
404230   2  1        ELSE                                                        XECO4230
                        RETURN (':FALSE');                                       XECO4240
404250   2  1     WHEN ('OR')                                                    XECO4250
                     IF ITEM1 = ':TRUE' ¦ ITEM2 = ':TRUE' THEN                   XECO4260
                        RETURN (':TRUE');                                        XECO4270
404280   2  1        ELSE                                                        XECO4280
                        RETURN (':FALSE');                                       XECO4290
404300   2  1     WHEN ('XOR')                                                   XECO4300
                     IF (ITEM1 = ':TRUE' & ITEM2 = ':FALSE')                     XECO4310
                        ¦ (ITEM1 = ':FALSE' & ITEM2 = ':TRUE') THEN              XECO4320
                        RETURN (':TRUE');                                        XECO4330
404340   2  1        ELSE                                                        XECO4340
                        RETURN (':FALSE');                                       XECO4350
404360   2  1     WHEN ('+')                                                     XECO4360
                     RETURN (CHAR (FIXED (ITEM1) + FIXED (ITEM2)));              XECO4370
404380   2  1     WHEN ('-')                                                     XECO4380
                     RETURN (CHAR (FIXED (ITEM1) - FIXED (ITEM2)));              XECO4390
404400   2  1     WHEN ('*')                                                     XECO4400
                     RETURN (CHAR (FIXED (ITEM1) * FIXED (ITEM2)));              XECO4410
404420   2  1     WHEN ('/')                                                     XECO4420
                     RETURN (CHAR (FIXED (FIXED (ITEM1) / FIXED (ITEM2))));      XECO4430
404440   2  1     WHEN ('!')                                                     XECO4440
                     RETURN (CHAR (FIXED (ITEM1) ** FIXED (ITEM2)));             XECO4450
404460   2  1     WHEN ('¦')                                                     XECO4460
                     RETURN (ITEM1 ¦¦ ITEM2);                                    XECO4470
404480   2  1     OTHERWISE DO;                                                  XECO4480
404490   2  2        PUT FILE (ERROR) SKIP EDIT                                  XECO4490
                        ('BINARY OPERATOR NOT SUPPORTED =$',REL,'$$',            XECO4500
```

125

```
 NUMBER LEV NT

                         'ITEM1 ARGUMENT =$',ITEM1,'$$',                    XEC04510
                         'ITEM2 ARGUMENT =$',ITEM2,'$$',                    XEC04520
                         'FORCED RESULT =$:EMPTY$$')                        XEC04530
                        (3 (SKIP,A,A,A),SKIP,A);                            XEC04540
 404550    2  2      RETURN (':EMPTY');                                     XEC04550
 404560    2  2      END;                                                   XEC04560
 404570    2  1    END;                                                     XEC04570
```

```
NUMBER LEV NT
                                                                    XEC04580
404580   2  0  MESS:                                                XEC04590
               PUT FILE (ERROR) SKIP EDIT                           XEC04600
                  ('CONV/OVFLOW/BAD_ARG ERROR FOR BINARY_OP',       XEC04610
                   'ITEM1 ARGUMENT =$',ITEM1,'$$',                  XEC04620
                   'ITEM2 ARGUMENT =$',ITEM2,'$$',                  XEC04630
                   'MIXED BINARY OPERATOR =$',REL,'$$',             XEC04640
                   'FORCED RESULT =$:EMPTY$$')                      XEC04650
                  (A,3 (SKIP,A,A,A),SKIP,A);                        XEC04660
404660   2  0  RETURN (':EMPTY');                                   XEC04670
                                                                    XEC04680
404680   2  0  END BINARY;                                          XEC04690
```

```
NUMBER LEV NT

404700   1  0  CREATE_LIST: PROC (LIST,USE_NUM) RETURNS (BIT (1));                     XECO4700
                                                                                        XECO4710
               /*MODIFIES LIST*/                                                        XECO4720
404730   2  0  DCL (LIST,P,Q,TAILQ) PTR,                                                XECO4730
                    (USE_NUM,N) FIXED;                                                  XECO4740
                                                                                        XECO4750
404760   2  0  IF ^ENTITY (CUR_ES).ATTR (USE_NUM).SING_OCC THEN DO;                     XECO4760
404770   2  1     PUT FILE (ERROR) SKIP EDIT                                            XECO4770
                     ('ATTEPT TO OPERATE ON NON-SING_OCC ITEM',                         XECO4780
                      'ITEM ARGUMENT =$',ENTITY (CUR_ES).ATTR (USE_NUM).USES,'$$',      XECO4790
                      'FORCED RESULT =$:EMPTY$$')                                       XECO4800
                     (A,SKIP,A,A,A,SKIP,A);                                             XECO4810
404820   2  1     RETURN (CREATE_CONST (LIST,':EMPTY'));                                XECO4820
404830   2  1     END;                                                                  XECO4830
404840   2  0  LIST = NULL();                                                           XECO4840
404850   2  0  N = 0;                                                                   XECO4850
404860   2  0  P = ENTITY (CUR_ES).ATTR (USE_NUM).LIST;                                 XECO4860
404870   2  0  DO WHILE (P ^= NULL ());                                                 XECO4870
404880   2  1     N = N + 1;                                                            XECO4880
404890   2  1     ALLOCATE ATTRIB SET (Q);                                              XECO4890
404900   2  1     Q -> ATTRIB.ITEM = P -> ATTRIB.ITEM;                                  XECO4900
404910   2  1     IF LIST = NULL () THEN                                                XECO4910
                     LIST = Q;                                                          XECO4920
404930   2  1     ELSE                                                                  XECO4930
                     TAILQ -> ATTRIB.NEXT = Q;                                          XECO4940
404950   2  1     TAILQ = Q;                                                            XECO4950
404960   2  1     P = P -> ATTRIB.NEXT;                                                 XECO4960
404970   2  1     END;                                                                  XECO4970
404980   2  0  TAILQ -> ATTRIB.NEXT = NULL ();                                          XECO4980
404990   2  0  RETURN ('1'B);                                                           XECO4990
                                                                                        XECO5000
405010   2  0  END CREATE_LIST;                                                         XECO5010
                                                                                        XECO5020
```

```
NUMBER LEV NT

405030   1  0  CREATE_CONST: PROC (LIST,CONSTNM) RETURNS (BIT (1));          XEC05030
                                                                             XEC05040
               /*MODIFIES LIST*/                                             XEC05050
405060   2  0  DCL LIST PTR,                                                 XEC05060
                    CONSTNM CHAR (30) VAR;                                   XEC05070
                                                                             XEC05080
405090   2  0  ALLOCATE ATTRIB SET (LIST);                                   XEC05090
405100   2  0  LIST -> ATTRIB.ITEM = CONSTNM;                                XEC05100
405110   2  0  LIST -> ATTRIB.NEXT = NULL ();                                XEC05110
405120   2  0  RETURN ('0'B);                                                XEC05120
                                                                             XEC05130
405140   2  0  END CREATE_CONST;                                             XEC05140
                                                                             XEC05150
```

```
 NUMBER LEV NT

 405160   1  0  TMPLTE: PROC (ITEM,TEMP_ITEM) RETURNS (BIT (1)) RECURSIVE;      XECO5160
                                                                                XECO5170
 405180   2  0  DCL (ITEM,TEMP_ITEM) CHAR (*) VAR,                              XECO5180
                      I FIXED;                                                  XECO5190
                                                                                XECO5200
 405210   2  0  IF TEMP_ITEM = '' THEN                                          XECO5210
                   IF ITEM = '' THEN                                            XECO5220
                      RETURN ('1'B);                                            XECO5230
 405240   2  0     ELSE                                                         XECO5240
                      RETURN ('0'B);                                            XECO5250
 405260   2  0  ELSE IF SUBSTR (TEMP_ITEM,1,1) = '¦' THEN                       XECO5260
                   IF ITEM = '' & LENGTH (TEMP_ITEM) = 1 THEN                   XECO5270
                      RETURN ('1'B);                                            XECO5280
 405290   2  0     ELSE DO;                                                     XECO5290
 405300   2  1        DO I = 1 TO LENGTH (ITEM) + 1;                            XECO5300
 405310   2  2           IF TMPLTE (SUBSTR (ITEM,I),SUBSTR (TEMP_ITEM,2)) THEN  XECO5310
                            RETURN ('1'B);                                      XECO5320
 405330   2  2           END;                                                   XECO5330
 405340   2  1        RETURN ('0'B);                                            XECO5340
 405350   2  1        END;                                                      XECO5350
 405360   2  0  ELSE IF ITEM = '' THEN                                          XECO5360
                   RETURN ('0'B);                                               XECO5370
 405380   2  0  ELSE IF SUBSTR (TEMP_ITEM,1,1) = '~' THEN                       XECO5380
                   RETURN (TMPLTE (SUBSTR (ITEM,2),SUBSTR (TEMP_ITEM,2)));      XECO5390
 405400   2  0  ELSE                                                            XECO5400
                   RETURN (SUBSTR (ITEM,1,1) = SUBSTR (TEMP_ITEM,1,1)           XECO5410
                           & TMPLTE (SUBSTR (ITEM,2),SUBSTR (TEMP_ITEM,2)));    XECO5420
                                                                                XECO5430
 405440   2  0  END TMPLTE;                                                     XECO5440
                                                                                XECO5450
```

```
 NUMBER LEV NT

 405460   1  0  DISPOSE: PROC (LIST);                                XECO5460
                                                                     XECO5470
 405480   2  0  DCL (LIST,PNOW,PLAST) PTR;                           XECO5480
                                                                     XECO5490
 405500   2  0  PNOW = LIST;                                         XECO5500
 405510   2  0  DO WHILE (PNOW ^= NULL);                             XECO5510
 405520   2  1     PLAST = PNOW;                                     XECO5520
 405530   2  1     PNOW = PNOW -> ATTRIB.NEXT;                       XECO5530
 405540   2  1     FREE PLAST -> ATTRIB;                             XECO5540
 405550   2  1     END;                                              XECO5550
 405560   2  0  LIST = NULL ();                                      XECO5560
                                                                     XECO5570
 405580   2  0  END DISPOSE;                                         XECO5580
                                                                     XECO5590
```

```
NUMBER LEV NT

405600   1  0  MUNION: PROC;                                                  XECO5600
                                                                              XECO5610
405620   2  0  DCL (I,J) FIXED,                                               XECO5620
                    STOP BIT (1);                                             XECO5630
                                                                              XECO5640
405650   2  0  COPY_COUNT = 0;                                                XECO5650
405660   2  0  CALL LOCATEKEYS;                                               XECO5660
405670   2  0  DO I = 1, 2;                                                   XECO5670
405680   2  1     DO J = 1 TO 15;                                             XECO5680
405690   2  2        ATAIL (I,J) = ENTITY (ENTITY (CUR_ES).N_PARENT (I))      XECO5690
                                   .ATTR (J).LIST;                            XECO5700
405710   2  2        END;                                                     XECO5710
405720   2  1     END;                                                        XECO5720
405730   2  0  ATAIL (0,*) = NULL ();                                         XECO5730
405740   2  0  NEWLVL (*,*) = 0;                                              XECO5740
405750   2  0  DO WHILE (ATAIL (1,1) ^= NULL ());                             XECO5750
405760   2  1     CALL COPYSET (ENTITY (CUR_ES).N_PARENT (1),                 XECO5760
                                1,                                            XECO5770
                                ENTITY (CUR_ES).VES_PAR -> N_MAP (1).NUM (*)); XECO5780
405790   2  1     END;                                                        XECO5790
405800   2  0  DO WHILE (ATAIL (2,1) ^= NULL ());                             XECO5800
405810   2  1     DO I = 1 TO 15;                                             XECO5810
405820   2  2        ATAIL (1,I) = ENTITY (ENTITY (CUR_ES).N_PARENT (1))      XECO5820
                                   .ATTR (I).LIST;                            XECO5830
405840   2  2        END;                                                     XECO5840
405850   2  1     NEWLVL (1,*) = 0;                                           XECO5850
405860   2  1     STOP = '0'B;                                                XECO5860
405870   2  1     DO WHILE (^STOP & ATAIL (1,1) ^= NULL ());                  XECO5870
405880   2  2        STOP = IUMATCH (ENTITY (CUR_ES).VES_PAR -> N_MAP);       XECO5880
405890   2  2        CALL SKIPSET (ENTITY (CUR_ES).N_PARENT (1),1);           XECO5890
405900   2  2        END;                                                     XECO5900
405910   2  1     IF STOP THEN                                                XECO5910
                     CALL SKIPSET (ENTITY (CUR_ES).N_PARENT (2),2);           XECO5920
405930   2  1     ELSE                                                        XECO5930
                     CALL COPYSET (ENTITY (CUR_ES).N_PARENT (2),              XECO5940
                                   2,                                         XECO5950
                                   ENTITY (CUR_ES).VES_PAR -> N_MAP (2).NUM (*)); XECO5960
405970   2  1     END;                                                        XECO5970
405980   2  0  ENTITY (CUR_ES).DEPTH = COPY_COUNT;                            XECO5980
                                                                              XECO5990
406000   2  0  END MUNION;                                                    XECO6000
                                                                              XECO6010
                                                                              XECO6020
```

```
NUMBER LEV NT

406030   1  0  MINTER: PROC;                                                         XEC06030
                                                                                     XEC06040
406050   2  0  DCL I FIXED,                                                          XEC06050
                   STOP BIT (1);                                                     XEC06060
                                                                                     XEC06070
406080   2  0  COPY_COUNT = 0;                                                       XEC06080
406090   2  0  CALL LOCATEKEYS;                                                      XEC06090
406100   2  0  DO I = 1 TO 15;                                                       XEC06100
406110   2  1     ATAIL (1,I) = ENTITY (ENTITY (CUR_ES).N_PARENT (1)).ATTR (I).LIST; XEC06110
406120   2  1     END;                                                               XEC06120
406130   2  0  ATAIL (0,*) = NULL ();                                                XEC06130
406140   2  0  NEWLVL (*,*) = 0;                                                     XEC06140
406150   2  0  DO WHILE (ATAIL (1,1) ^= NULL ());                                    XEC06150
406160   2  1     DO I = 1 TO 15;                                                    XEC06160
406170   2  2        ATAIL (2,I) = ENTITY (ENTITY (CUR_ES).N_PARENT (2))             XEC06170
                                   .ATTR (I).LIST;                                   XEC06180
406190   2  2        END;                                                            XEC06190
406200   2  1     NEWLVL (2,*) = 0;                                                  XEC06200
406210   2  1     STOP = '0'B;                                                       XEC06210
406220   2  1     DO WHILE (^STOP & ATAIL (1,1) ^= NULL ());                         XEC06220
406230   2  2        STOP = IUMATCH (ENTITY (CUR_ES).VES_PAR -> N_MAP);              XEC06230
406240   2  2        CALL SKIPSET (ENTITY (CUR_ES).N_PARENT (2),2);                  XEC06240
406250   2  2        END;                                                            XEC06250
406260   2  1     IF STOP THEN                                                       XEC06260
                     CALL COPYSET (ENTITY (CUR_ES).N_PARENT (1),                     XEC06270
                                   1,                                                XEC06280
                                   ENTITY (CUR_ES).VES_PAR -> N_MAP (1).NUM (*));    XEC06290
406300   2  1     ELSE                                                               XEC06300
                     CALL SKIPSET (ENTITY (CUR_ES).N_PARENT (1),1);                  XEC06310
406320   2  1     END;                                                               XEC06320
406330   2  0  ENTITY (CUR_ES).DEPTH = COPY_COUNT;                                   XEC06330
                                                                                     XEC06340
406350   2  0  END MINTER;                                                           XEC06350
                                                                                     XEC06360
                                                                                     XEC06370
```

```
 NUMBER LEV NT

 406380   1  0  SINGLE: PROC;                                                      XECO6380
                                                                                   XECO6390
 406400   2  0  DCL 1 ENTMAP (2),                                                  XECO6400
                       2 NUM (15) FIXED;                                           XECO6410
 406420   2  0  DCL (I,J) FIXED,                                                   XECO6420
                    STOP BIT (1);                                                  XECO6430
                                                                                   XECO6440
 406450   2  0  COPY_COUNT = 0;                                                    XECO6450
 406460   2  0  ENTITY (0) = ENTITY (CUR_ES);                                      XECO6460
 406470   2  0  ENTITY (CUR_ES).ATTR (*).LIST = NULL ();                           XECO6470
 406480   2  0  DO I = 1 TO 15;                                                    XECO6480
 406490   2  1     ENTMAP (*).NUM (I) = I;                                         XECO6490
 406500   2  1     ATAIL (1,I) = ENTITY (0).ATTR (I).LIST;                         XECO6500
 406510   2  1     END;                                                            XECO6510
 406520   2  0  ATAIL (0,*) = NULL ();                                             XECO6520
 406530   2  0  NEWLVL (*,*) = 0;                                                  XECO6530
 406540   2  0  DO WHILE (ATAIL (1,1) ^= NULL ());                                 XECO6540
 406550   2  1     IF ATAIL (1,1) -> ATTRIB.ITEM ^= ':EMPTYFILL' THEN DO;          XECO6550
 406560   2  2        ATAIL (2,*) = ATAIL (1,*);                                   XECO6560
 406570   2  2        NEWLVL (2,*) = NEWLVL (1,*);                                 XECO6570
 406580   2  2        CALL SKIPSET (0,2);                                          XECO6580
 406590   2  2        STOP = '0'B;                                                 XECO6590
 406600   2  2        DO WHILE (^STOP & ATAIL (2,1) ^= NULL ());                   XECO6600
 406610   2  3           IF ATAIL (2,1) -> ATTRIB.ITEM ^= ':EMPTYFILL' THEN        XECO6610
                            STOP = IUMATCH (ENTMAP);                               XECO6620
 406630   2  3           CALL SKIPSET (0,2);                                       XECO6630
 406640   2  3           END;                                                      XECO6640
 406650   2  2        IF STOP THEN                                                 XECO6650
                         ATAIL (1,1) -> ATTRIB.ITEM = ':EMPTYFILL';                XECO6660
 406670   2  2        END;                                                         XECO6670
 406680   2  1     CALL SKIPSET (0,1);                                             XECO6680
 406690   2  1     END;                                                            XECO6690
 406700   2  0  DO I = 1 TO 15;                                                    XECO6700
 406710   2  1     ATAIL (1,I) = ENTITY (0).ATTR (I).LIST;                         XECO6710
 406720   2  1     END;                                                            XECO6720
 406730   2  0  NEWLVL (1,*) = 0;                                                  XECO6730
 406740   2  0  DO WHILE (ATAIL (1,1) ^= NULL ());                                 XECO6740
 406750   2  1     IF ATAIL (1,1) -> ATTRIB.ITEM = ':EMPTYFILL' THEN               XECO6750
                      CALL SKIPSET (0,1);                                          XECO6760
 406770   2  1     ELSE                                                            XECO6770
                      CALL COPYSET (0,                                             XECO6780
                                    1,                                             XECO6790
                                    ENTMAP (1).NUM (*));                           XECO6800
 406810   2  1     END;                                                            XECO6810
 406820   2  0  DO I = 1 TO 15 WHILE (ENTITY (0).ATTR (I).LIST ^= NULL ());        XECO6820
 406830   2  1     CALL DISPOSE (ENTITY (0).ATTR (I).LIST);                        XECO6830
 406840   2  1     END;                                                            XECO6840
 406850   2  0  ENTITY (CUR_ES).DEPTH = COPY_COUNT;                                XECO6850
                                                                                   XECO6860
```

```
NUMBER LEV NT

406870   2  0  END SINGLE;                                          XEC06870
                                                                    XEC06880
                                                                    XEC06890
```

```
NUMBER LEV NT

406900   1  0  CRTESN. PROC;                                                                 XECO6900
                                                                                             XECO6910
406920   2  0  DCL (I,N,REPEATVAL) FIXED;                                                    XECO6920
                                                                                             XECO6930
406940   2  0  COPY_COUNT = 0;                                                               XECO6940
406950   2  0  DO I = 1 TO 15;                                                               XECO6950
406960   2  1     ATAIL (2,I) = ENTITY (ENTITY (CUR_ES).N_PARENT (1)).ATTR (I).LIST;         XECO6960
406970   2  1     END;                                                                       XECO6970
406980   2  0  ATAIL (0,*) = NULL;                                                           XECO6980
406990   2  0  NEWLVL (*,*) = 0;                                                             XECO6990
407000   2  0  REPEATVAL = ENTITY (ENTITY (CUR_ES).N_PARENT (2)).DEPTH;                      XECO7000
407010   2  0  DO WHILE (ATAIL (2,1) ^= NULL ());                                            XECO7010
407020   2  1     DO N = 1 TO REPEATVAL;                                                     XECO7020
407030   2  2        ATAIL (1,*) = ATAIL (2,*);                                              XECO7030
407040   2  2        NEWLVL (1,*) = 0;                                                       XECO7040
407050   2  2        CALL COPYSET (ENTITY (CUR_ES).N_PARENT (1),                             XECO7050
                                   1,                                                        XECO7060
                                   ENTITY (CUR_ES).VES_PAR -> N_MAP (1).NUM (*));            XECO7070
407080   2  2        END;                                                                    XECO7080
407090   2  1     CALL SKIPSET (ENTITY (CUR_ES).N_PARENT (1),2);                             XECO7090
407100   2  1     END;                                                                       XECO7100
407110   2  0  ENTITY (CUR_ES).DEPTH = COPY_COUNT;                                           XECO7110
407120   2  0  DO I = 1 TO 15;                                                               XECO7120
407130   2  1     ATAIL (1,I) = ENTITY (ENTITY (CUR_ES).N_PARENT (2)).ATTR (I).LIST;         XECO7130
407140   2  1     END;                                                                       XECO7140
407150   2  0  ATAIL (0,*) = NULL ();                                                        XECO7150
407160   2  0  NEWLVL (*,*) = 0;                                                             XECO7160
407170   2  0  REPEATVAL = ENTITY (ENTITY (CUR_ES).N_PARENT (1)).DEPTH;                      XECO7170
407180   2  0  DO N = 1 TO REPEATVAL;                                                        XECO7180
407190   2  1     ATAIL (2,*) = ATAIL (1,*);                                                 XECO7190
407200   2  1     NEWLVL (2,*) = 0;                                                          XECO7200
407210   2  1     DO WHILE (ATAIL (2,1) ^= NULL ());                                         XECO7210
407220   2  2        CALL COPYSET (ENTITY (CUR_ES).N_PARENT (2),                             XECO7220
                                   2,                                                        XECO7230
                                   ENTITY (CUR_ES).VES_PAR -> N_MAP (2).NUM (*));            XECO7240
407250   2  2        END;                                                                    XECO7250
407260   2  1     END;                                                                       XECO7260
                                                                                             XECO7270
407280   2  0  END CRTESN;                                                                   XECO7280
                                                                                             XECO7290
                                                                                             XECO7300
```

```
NUMBER LEV NT

407310   1  0  SKIPSET: PROC (SK_ENT,ENT_NUM);                                           XECO7310
                                                                                         XECO7320
407330   2  0  DCL (SK_ENT,ENT_NUM,I,LVL_LIM) FIXED;                                     XECO7330
                                                                                         XECO7340
407350   2  0  DO I = 1 TO 15 WHILE (ENTITY (SK_ENT).ATTR (I).USES ^= '');               XECO7350
407360   2  1     IF ENTITY (SK_ENT).ATTR (I).SING_OCC THEN                              XECO7360
                     LVL_LIM = NEWLVL (ENT_NUM,I) + 1;                                   XECO7370
407380   2  1     ELSE                                                                   XECO7380
                     LVL_LIM = NEWLVL (ENT_NUM,ENTITY (SK_ENT).ATTR (I).A_PARENT);       XECO7390
407400   2  1     DO WHILE (ATAIL (ENT_NUM,I) ^= NULL ()                                 XECO7400
                            & ATAIL (ENT_NUM,I) -> ATTRIB.LEVEL <= LVL_LIM);             XECO7410
407420   2  2        NEWLVL (ENT_NUM,I) = NEWLVL (ENT_NUM,I) + 1;                        XECO7420
407430   2  2        ATAIL (ENT_NUM,I) = ATAIL (ENT_NUM,I) -> ATTRIB.NEXT;               XECO7430
407440   2  2        END;                                                                XECO7440
407450   2  1     END;                                                                   XECO7450
                                                                                         XECO7460
407470   2  0  END SKIPSET;                                                              XECO7470
                                                                                         XECO7480
```

```
NUMBER LEV NT

407490   1  0  COPYSET: PROC (PAR_ENT,PAR_NUM,MAP_NUM);

407510   2  0  DCL (PAR_ENT,PAR_NUM,I,LVL_ST,LVL_LIM,OLDLVL (0:2,15)) FIXED;
407520   2  0  DCL MAP_NUM (15) FIXED;
407530   2  0  DCL P PTR;

407550   2  0  OLDLVL (*,*) = NEWLVL (*,*);
407560   2  0  DO I = 1 TO 15 WHILE (ENTITY (CUR_ES).ATTR (I).USES ^= '');
407570   2  1     IF MAP_NUM (I) > 0
                     & ENTITY (PAR_ENT).ATTR (MAP_NUM (I)).SING_OCC THEN DO;
407590   2  2        LVL_ST = OLDLVL (PAR_NUM,MAP_NUM (I));
407600   2  2        LVL_LIM = NEWLVL (PAR_NUM,MAP_NUM (I)) + 1;
407610   2  2        END;
407620   2  1     ELSE DO;
407630   2  2        LVL_ST = OLDLVL (PAR_NUM,
                                 ENTITY (PAR_ENT).ATTR (MAP_NUM (I)).A_PARENT);
407650   2  2        LVL_LIM = NEWLVL (PAR_NUM,
                                  ENTITY (PAR_ENT).ATTR (MAP_NUM (I)).A_PARENT);
407670   2  2        END;
407680   2  1     DO WHILE (ATAIL (PAR_NUM,MAP_NUM (I)) ^= NULL ()
                        & ATAIL (PAR_NUM,MAP_NUM (I)) -> ATTRIB.LEVEL <= LVL_LIM);
407700   2  2        NEWLVL (0,I) = NEWLVL (0,I) + 1;
407710   2  2        NEWLVL (PAR_NUM,MAP_NUM (I)) = NEWLVL (PAR_NUM,MAP_NUM (I)) + 1;
407720   2  2        ALLOCATE ATTRIB SET (P);
407730   2  2        P -> ATTRIB.ITEM = ATAIL (PAR_NUM,MAP_NUM (I)) -> ATTRIB.ITEM;
407740   2  2        P -> ATTRIB.LEVEL = OLDLVL (0,I)
                                   + ATAIL (PAR_NUM,MAP_NUM (I)) -> ATTRIB.LEVEL
                                   - LVL_ST;
407770   2  2        IF ENTITY (CUR_ES).ATTR (I).LIST = NULL () THEN
                        ENTITY (CUR_ES).ATTR (I).LIST = P;
407790   2  2        ELSE
                        ATAIL (0,I) -> ATTRIB.NEXT = P;
407810   2  2        ATAIL (0,I) = P;
407820   2  2        ATAIL (PAR_NUM,MAP_NUM (I)) = ATAIL (PAR_NUM,MAP_NUM (I))
                                                   -> ATTRIB.NEXT;
407840   2  2        END;
407850   2  1     END;
407860   2  0  COPY_COUNT = COPY_COUNT + 1;

407880   2  0  END COPYSET;
```

135

```
NUMBER LEV NT

407900   1  0  COPYALL: PROC (FR_ENT);                                                    XEC07900
                                                                                          XEC07910
407920   2  0  DCL (FR_ENT,I,J) FIXED;                                                    XEC07920
                                                                                          XEC07930
407940   2  0  ENTITY (CUR_ES).DEPTH = ENTITY (FR_ENT).DEPTH;                             XEC07940
407950   2  0  ENTITY (CUR_ES).ATTR (*).VES_KEY = ENTITY (FR_ENT).ATTR (*).VES_KEY;       XEC07950
407960   2  0  ENTITY (CUR_ES).ATTR (*).CART_KEY = ENTITY (FR_ENT).ATTR (*).CART_KEY;     XEC07960
407970   2  0  ENTITY (CUR_ES).ATTR (*).SING_OCC = ENTITY (FR_ENT).ATTR (*).SING_OCC;     XEC07970
407980   2  0  ENTITY (CUR_ES).ATTR (*).A_PARENT = ENTITY (FR_ENT).ATTR (*).A_PARENT;     XEC07980
407990   2  0  ENTITY (CUR_ES).ATTR (*).USES = ENTITY (FR_ENT).ATTR (*).USES;             XEC07990
408000   2  0  DO I = 1 TO 15 WHILE (ENTITY (FR_ENT).ATTR (I).USES ^= '');                XEC08000
408010   2  1     Q = ENTITY (FR_ENT).ATTR (I).LIST;                                      XEC08010
408020   2  1     DO WHILE (Q ^= NULL ());                                                XEC08020
408030   2  2        ALLOCATE ATTRIB SET (P);                                             XEC08030
408040   2  2        P -> ATTRIB.ITEM = Q -> ATTRIB.ITEM;                                 XEC08040
408050   2  2        P -> ATTRIB.LEVEL = Q -> ATTRIB.LEVEL;                               XEC08050
408060   2  2        IF ENTITY (CUR_ES).ATTR (I).LIST = NULL () THEN                      XEC08060
                        ENTITY (CUR_ES).ATTR (I).LIST = P;                                XEC08070
408080   2  2        ELSE                                                                 XEC08080
                        TAILP -> ATTRIB.NEXT = P;                                         XEC08090
408100   2  2        TAILP = P;                                                           XEC08100
408110   2  2        Q = Q -> ATTRIB.NEXT;                                                XEC08110
408120   2  2        END;                                                                 XEC08120
408130   2  1     TAILP -> ATTRIB.NEXT = NULL ();                                         XEC08130
408140   2  1     END;                                                                    XEC08140
                                                                                          XEC08150
408160   2  0  END COPYALL;                                                               XEC08160
                                                                                          XEC08170
```

```
NUMBER LEV NT

408180   1  0  IUMATCH: PROC (ENTMAP) RETURNS (BIT (1));                       XECO8180
                                                                               XECO8190
408200   2  0  DCL 1 ENTMAP (2),                                               XECO8200
                      2 NUM (15) FIXED;                                        XECO8210
                                                                               XECO8220
408230   2  0  DO I = 1 TO 15 WHILE ((KEYSET (I)) ^= O);                       XECO8230
408240   2  1     IF ATAIL (1,ENTMAP (1).NUM (KEYSET (I))) -> ATTRIB.ITEM      XECO8240
                     ^= ATAIL (2,ENTMAP (2).NUM (KEYSET (I))) -> ATTRIB.ITEM THEN XECO8250
                     RETURN ('O'B);                                            XECO8260
408270   2  1     END;                                                         XECO8270
408280   2  0  RETURN ('1'B);                                                  XECO8280
                                                                               XECO8290
408300   2  0  END IUMATCH;                                                    XECO8300
                                                                               XECO8310
```

```
 NUMBER LEV NT

                                                                             XEC08320
 408320   1  0  LOCATEKEYS: PROC;                                            XEC08330
                                                                             XEC08340
 408340   2  0  DCL (I,J) FIXED;                                             XEC08350
                                                                             XEC08360
 408360   2  0  KEYSET (*) = 0;                                              XEC08370
 408370   2  0  J = 0;                                                       XEC08380
 408380   2  0  DO I = 1 TO 15 WHILE (ENTITY (CUR_ES).ATTR (I).USES ^= '');  XEC08390
 408390   2  1     IF ENTITY (CUR_ES).ATTR (I).VES_KEY THEN DO;              XEC08400
 408400   2  2        J = J + 1;                                             XEC08410
 408410   2  2        KEYSET (J) = I;                                        XEC08420
 408420   2  2        END;                                                   XEC08430
 408430   2  1     END;                                                      XEC08440
                                                                             XEC08450
 408450   2  0  END LOCATEKEYS;                                              XEC08460
```

```
NUMBER LEV NT

408470   1  0  DECRSET: PROC;                                                    XECO8470
                                                                                 XECO8480
408490   2  0  DCL ENTMAP_NUM (15) FIXED;                                        XECO8490
408500   2  0  DCL I FIXED;                                                      XECO8500
408510   2  0  DCL P PTR;                                                        XECO8510
                                                                                 XECO8520
408530   2  0  COPY_COUNT = 0;                                                   XECO8530
408540   2  0  ENTITY (0) = ENTITY (CUR_ES);                                     XECO8540
408550   2  0  ENTITY (CUR_ES).ATTR (*).LIST = NULL ();                          XECO8550
408560   2  0  DO I = 1 TO 15;                                                   XECO8560
408570   2  1     ENTMAP_NUM (I) = I;                                            XECO8570
408580   2  1     ATAIL (1,I) = ENTITY (0).ATTR (I).LIST;                        XECO8580
408590   2  1     END;                                                           XECO8590
408600   2  0  ATAIL (0,*) = NULL ();                                            XECO8600
408610   2  0  NEWLVL (*,*) = 0;                                                 XECO8610
408620   2  0  P = SLCTLIST;                                                     XECO8620
408630   2  0  DO WHILE (P ^= NULL ());                                          XECO8630
408640   2  1     IF P -> ATTRIB.ITEM = ':TRUE' THEN                             XECO8640
                     CALL COPYSET (0,                                            XECO8650
                                   1,                                            XECO8660
                                   ENTMAP_NUM (*));                              XECO8670
408680   2  1     ELSE                                                           XECO8680
                     CALL SKIPSET (0,1);                                         XECO8690
408700   2  1     P = P -> ATTRIB.NEXT;                                          XECO8700
408710   2  1     END;                                                           XECO8710
408720   2  0  DO I = 1 TO 15 WHILE (ENTITY (0).ATTR (I).LIST ^= NULL ());       XECO8720
408730   2  1     CALL DISPOSE (ENTITY (0).ATTR (I).LIST);                       XECO8730
408740   2  1     END;                                                           XECO8740
408750   2  0  ENTITY (CUR_ES).DEPTH = COPY_COUNT;                               XECO8750
                                                                                 XECO8760
408770   2  0  END DECRSET;                                                      XECO8770
                                                                                 XECO8780
```

142

```
NUMBER LEV NT
                                                                      XEC08790
408790   1  0  GETS: PROC (LIST,TERM_ITEM) RETURNS (BIT (1));         XEC08800
                                                                      XEC08810
408810   2  0  DCL LIST CHAR (*) VAR,                                 XEC08820
                   TERM_ITEM CHAR (1),                                XEC08830
                   RTN_LIST CHAR (30) VAR,                            XEC08840
                   I FIXED;                                           XEC08850
                                                                      XEC08860
408860   2  0  I = INDEX (LIST,TERM_ITEM);                            XEC08870
408870   2  0  IF I = O THEN DO;                                      XEC08880
408880   2  1     RTN_LIST = LIST;                                    XEC08890
408890   2  1     LIST = '';                                          XEC08900
408900   2  1     END;                                                XEC08910
408910   2  0  ELSE DO;                                               XEC08920
408920   2  1     RTN_LIST = SUBSTR (LIST,1,I - 1);                   XEC08930
408930   2  1     LIST = SUBSTR (LIST,I + 1);                         XEC08940
408940   2  1     END;                                                XEC08950
408950   2  0  RETURN (RTN_LIST);                                     XEC08960
                                                                      XEC08970
408970   2  0  END GETS;                                              XEC08980
```

```
NUMBER LEV NT

408990   1  0  END XECUTE;                                   XEC08990
```

DATE
ILME